Java Sequential File Series

//AccountRecord.java

// Java core packages

// @HW@

// add grade average point GPA

//

import java.io.Serializable;

public class AccountRecord implements Serializable { // if it is interface it

// should be implements

// (tagging interface);

private static final long serialVersionUID = 3417040053089159204L;

private int account;

private String firstName;

private String lastName;

//private String address;

//private String socsec;

//private double balance;

//private double gpa;

//private String title;

// no-argument constructor calls other constructor with

// default values

public AccountRecord() {

this(0, "", ""); // default way of private variables.

}

// initialize a record

public AccountRecord(int acct, String first, String last) {

setAccount(acct);

setFirstName(first);

setLastName(last);

}

// set account number

public void setAccount(int account) {

this.account = account;

}

// get account number

public int getAccount() {

return account;

}

// set first name

public void setFirstName(String first) {

firstName = first;

}

// get first name

public String getFirstName() {

return firstName;

}

// set last name

public void setLastName(String last) {

lastName = last;

}

// get last name

public String getLastName() {

return lastName;

}

} // end class AccountRecord

// BankUI.java

// Java core packages

import java.awt.\*;

import java.awt.event.KeyAdapter;

import java.awt.event.KeyEvent;

// Java extension packages

import javax.swing.\*;

public class BankUI extends JPanel {

private static final long serialVersionUID = 3417040053089159204L;

// label text for GUI

protected final static String names[] = { "Account number", "First name",

"Last name", "Address", "SocSec", "Balance", "GPA", "Title", "Transaction Amount" };

// GUI components; protected for future subclass access

protected JLabel labels[];

protected JTextField fields[];

protected JButton doTask1, doTask2;

protected JPanel innerPanelCenter, innerPanelSouth;

// number of text fields in GUI

protected int size;

// constants representing text fields in GUI

public static final int ACCOUNT = 0, FIRSTNAME = 1, LASTNAME = 2, ADDRESS = 3, SOCSEC = 4,

BALANCE = 5, GPA = 6, TITLE = 7, TRANSACTION = 8;

// Set up GUI. Constructor argument of 4 creates four rows

// of GUI components. Constructor argument of 5 (used in a

// later program) creates five rows of GUI components.

public BankUI(int mySize) {

size = mySize;

labels = new JLabel[size];

fields = new JTextField[size];

// create labels

for (int count = 0; count < labels.length; count++)

labels[count] = new JLabel(names[count]);

// create text fields

for (int count = 0; count < fields.length; count++)

fields[count] = new JTextField();

// set focus for text fields

for(int x=0; x < size; x++){

final int fieldSize = x;

fields[x].addKeyListener(new KeyAdapter(){

public void keyPressed(KeyEvent e){

if(e.getKeyCode() == KeyEvent.VK\_ENTER)

if(fieldSize == (size-1))

doTask2.requestFocus();

else fields[fieldSize+1].requestFocus();

}

});

}

// create panel to lay out labels and fields

innerPanelCenter = new JPanel();

innerPanelCenter.setLayout(new GridLayout(size, 2));

// attach labels and fields to innerPanelCenter

for (int count = 0; count < size; count++) {

innerPanelCenter.add(labels[count]);

innerPanelCenter.add(fields[count]);

}

// create generic buttons; no labels or event handlers

doTask1 = new JButton();

doTask2 = new JButton();

// create panel to lay out buttons and attach buttons

innerPanelSouth = new JPanel();

innerPanelSouth.add(doTask1);

innerPanelSouth.add(doTask2);

// set layout of this container and attach panels to it

setLayout(new BorderLayout());

add(innerPanelCenter, BorderLayout.CENTER);

add(innerPanelSouth, BorderLayout.SOUTH);

// validate layout

validate(); // make share panels fit.

// pack(); // make all panels in constant sizes

} // end constructor

// return reference to generic task button doTask1

public JButton getDoTask1Button() {

return doTask1;

}

// return reference to generic task button doTask2

public JButton getDoTask2Button() {

return doTask2;

}

// return reference to fields array of JTextFields

public JTextField[] getFields() {

return fields;

}

// clear content of text fields

public void clearFields() {

for (int count = 0; count < size; count++)

fields[count].setText("");

}

// set text field values; throw IllegalArgumentException if

// incorrect number of Strings in argument

public void setFieldValues(String strings[])

throws IllegalArgumentException {

if (strings.length != size)

throw new IllegalArgumentException("There must be " + size

+ " Strings in the array");

for (int count = 0; count < size; count++)

fields[count].setText(strings[count]);

}

// get array of Strings with current text field contents

public String[] getFieldValues() {

String values[] = new String[size];

for (int count = 0; count < size; count++)

values[count] = fields[count].getText();

return values; //&values[0]

}

} // end class BankUI

// CreateSequentialFile.java

// Java core packages

import java.io.\*;

import java.awt.\*;

import java.awt.event.\*;

// Java extension packages

import javax.swing.\*;

public class CreateSequentialFile extends JFrame {

private static final long serialVersionUID = 3417040053089159204L;

private ObjectOutputStream output;

private BankUI userInterface;

private JButton enterButton, openButton;

private JTextField setFcs;

// set up GUI

public CreateSequentialFile() {

super("Creating a Sequential File of Objects");

// create instance of reusable user interface

userInterface = new BankUI(3);

getContentPane().add(userInterface, BorderLayout.CENTER);

// get reference to generic task button doTask1 in BankUI

// and configure button for use in this program

openButton = userInterface.getDoTask1Button();

openButton.setText("Save into File ...");

openButton.setMnemonic('S');

// register listener to call openFile when button pressed

openButton.addActionListener(

// anonymous inner class to handle openButton event

new ActionListener() {

// call openFile when button pressed

public void actionPerformed(ActionEvent event) {

openFile();

}

} // end anonymous inner class

); // end call to addActionListener

// get reference to generic task button doTask2 in BankUI

// and configure button for use in this program

enterButton = userInterface.getDoTask2Button();

enterButton.setText("Enter");

enterButton.setEnabled(false); // disable button

enterButton.setMnemonic('E'); // hot key alt+E

// register listener to call addRecord when button pressed

enterButton.addActionListener(

// anonymous inner class to handle enterButton event

new ActionListener() {

// call addRecord when button pressed

public void actionPerformed(ActionEvent event) {

addRecord();

setFcs = userInterface.fields[0];

setFcs.requestFocus();

}

} // end anonymous inner class

); // end call to addActionListener

// add Enter key listener

enterButton.addKeyListener(new KeyAdapter() {

public void keyPressed(KeyEvent e) {

if (e.getKeyCode() == KeyEvent.VK\_ENTER)

addRecord();

setFcs = userInterface.fields[0];

setFcs.requestFocus();

} // keyPressed

}/\* KeyAdapter \*/);

// configure window

setDefaultCloseOperation(

WindowConstants.EXIT\_ON\_CLOSE );

// Get the size of the screen

Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();

int x = (((screenSize.width - this.getWidth()) / 2) / 2);

int y = (((screenSize.height - this.getHeight()) / 2) / 2);

this.setLocation(x, y);

// Set font style

Font myFont = new Font("Franklin", Font.BOLD, 14);

//produce the frame and show its

setSize( 500, 350 );

setResizable(false);

setVisible( true );

} // end CreateSequentialFile constructor

// allow user to specify file name

private void openFile() {

// display file dialog, so user can choose file to open

JFileChooser fileChooser = new JFileChooser();

fileChooser.setFileSelectionMode(JFileChooser.FILES\_ONLY);

int result = fileChooser.showSaveDialog(this);

// if user clicked Cancel button on dialog, return

if (result == JFileChooser.CANCEL\_OPTION) // JFileChooser.CANCEL\_OPTION

return;

// get selected file

File fileName = fileChooser.getSelectedFile(); // file treats the string

// \ into

// @"c:\java\date\r.dat" into "c:\\java\\date\\r.dat"

// display error if invalid

if (fileName == null || fileName.getName().equals(""))

JOptionPane.showMessageDialog(this, "Invalid File Name",

"Invalid File Name", JOptionPane.ERROR\_MESSAGE);

else {

// open file

try {

output = new ObjectOutputStream(new FileOutputStream(fileName));

openButton.setEnabled(false);

enterButton.setEnabled(true);

}

// process exceptions from opening file

catch (IOException ioException) {

JOptionPane.showMessageDialog(this, "Error Opening File",

"Error", JOptionPane.ERROR\_MESSAGE);

}

}

} // end method openFile

// close file and terminate application

private void closeFile() {

// close file

try {

output.close();

System.exit(0);

}

// process exceptions from closing file

catch (IOException ioException) {

JOptionPane.showMessageDialog(this, "Error closing file", "Error",

JOptionPane.ERROR\_MESSAGE);

System.exit(1);

}

}

// add record to file

public void addRecord() {

int accountNumber = 0;

AccountRecord record;

String fieldValues[] = userInterface.getFieldValues();

// if account field value is not empty

if (!fieldValues[BankUI.ACCOUNT].equals("")) {

// output values to file

try {

accountNumber = Integer.parseInt(fieldValues[BankUI.ACCOUNT]);

if (accountNumber > 0) {

// create new record

record = new AccountRecord(accountNumber,

fieldValues[BankUI.FIRSTNAME],

fieldValues[BankUI.LASTNAME]);

// output record and flush buffer

output.writeObject(record);

output.flush(); // flush make sure all the data are done

}

// clear textfields

userInterface.clearFields();

}

// process invalid account number or balance format

catch (NumberFormatException formatException) {

JOptionPane.showMessageDialog(this,

"Bad account number or balance",

"Invalid Number Format", JOptionPane.ERROR\_MESSAGE);

}

// process exceptions from file output

catch (IOException ioException) {

closeFile();

}

} // end if

} // end method addRecord

// execute application; CreateSequentialFile constructor

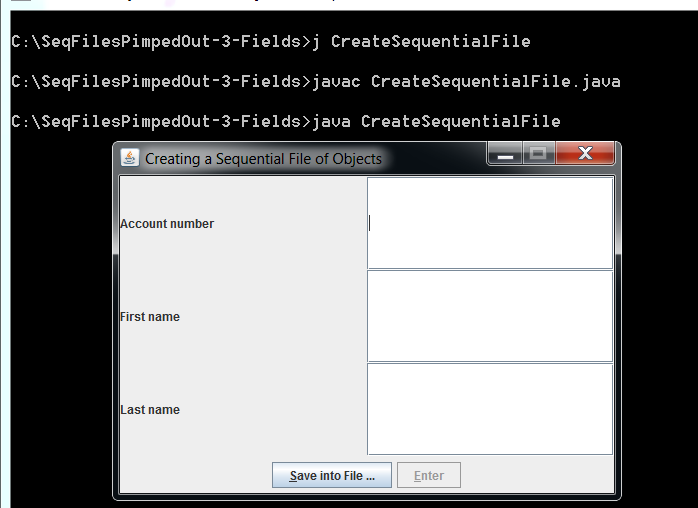
// displays window

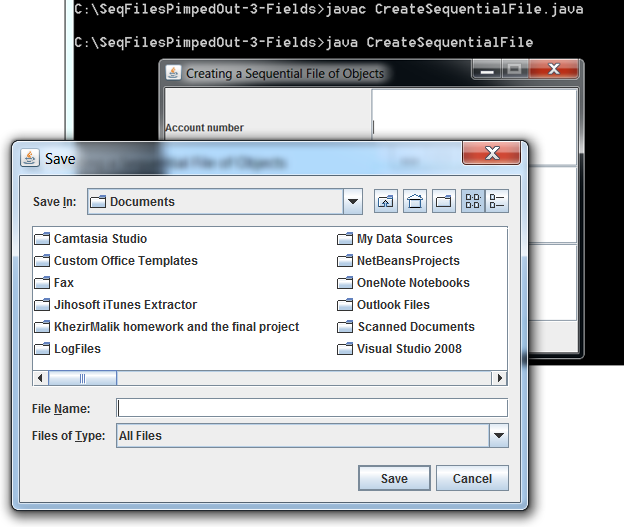
public static void main(String args[]) {

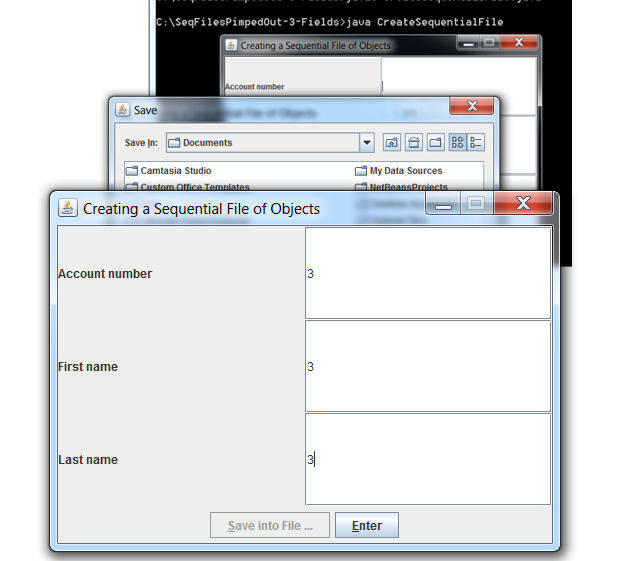
new CreateSequentialFile();

}

} // end class CreateSequentialFile







//ReadSquentialFile.java

// This program reads a file of objects sequentially

// and displays each record.

// Java core packages

import java.io.\*;

import java.awt.\*;

import java.awt.event.\*;

// Java extension packages

import javax.swing.\*;

@SuppressWarnings("serial")

public class ReadSequentialFile extends JFrame {

private static final long serialVersionUID = 3417040053089159204L;

private ObjectInputStream input;

private BankUI userInterface;

private JButton nextButton, openButton;

// Constructor -- initialize the Frame

public ReadSequentialFile()

{

super( "Reading a Sequential File of Objects" );

// create instance of reusable user interface

userInterface = new BankUI(3);

getContentPane().add(

userInterface, BorderLayout.CENTER );

for(int x = 0; x < 3; x++){

userInterface.fields[x].setEditable( false );

userInterface.fields[x].setBackground(Color.white);

}

// get reference to generic task button doTask1 from BankUI

openButton = userInterface.getDoTask1Button();

openButton.setText( "Open File" );

openButton.setMnemonic('o');

// register listener to call openFile when button pressed

openButton.addActionListener(

// anonymous inner class to handle openButton event

new ActionListener() {

// close file and terminate application

public void actionPerformed( ActionEvent event )

{

openFile();

}

} // end anonymous inner class

); // end call to addActionListener

// get reference to generic task button doTask2 from BankUI

nextButton = userInterface.getDoTask2Button();

nextButton.setText( "Next Record" );

nextButton.setEnabled( false );

nextButton.setMnemonic('N');

// register listener to call readRecord when button pressed

nextButton.addActionListener(

// anonymous inner class to handle nextRecord event

new ActionListener() {

// call readRecord when user clicks nextRecord

public void actionPerformed( ActionEvent event )

{

readRecord();

}

} // end anonymous inner class

); // end call to addActionListener

nextButton.addKeyListener(new KeyAdapter() {

public void keyPressed(KeyEvent e) {

if (e.getKeyCode() == KeyEvent.VK\_ENTER)

readRecord();

} // keyPressed

}/\*KeyAdapter\*/);

pack(); // pack the window, making it just big enough

// configure window

setDefaultCloseOperation(

WindowConstants.EXIT\_ON\_CLOSE );

// Get the size of the screen

Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();

int x = (((screenSize.width - this.getWidth()) / 2) / 2);

int y = (((screenSize.height - this.getHeight()) / 2) / 2);

this.setLocation(x, y);

// Set font style

Font myFont = new Font("Franklin", Font.BOLD, 14);

//produce the frame and show its

setSize( 500, 350 );

setResizable(false);

setVisible( true );

} // end ReadSequentialFile constructor

// enable user to select file to open

private void openFile()

{

// display file dialog so user can select file to open

JFileChooser fileChooser = new JFileChooser();

fileChooser.setFileSelectionMode(

JFileChooser.FILES\_ONLY );

int result = fileChooser.showOpenDialog( this );

// if user clicked Cancel button on dialog, return

if ( result == JFileChooser.CANCEL\_OPTION )

return;

// obtain selected file

File fileName = fileChooser.getSelectedFile();

// display error if file name invalid

if ( fileName == null ||

fileName.getName().equals( "" ) )

JOptionPane.showMessageDialog( this,

"Invalid File Name", "Invalid File Name",

JOptionPane.ERROR\_MESSAGE );

else {

// open file

try {

input = new ObjectInputStream(

new FileInputStream( fileName ) );

openButton.setEnabled( false );

nextButton.setEnabled( true );

nextButton.requestFocus();

}

// process exceptions opening file

catch ( IOException ioException ) {

JOptionPane.showMessageDialog( this,

"Error Opening File", "Error",

JOptionPane.ERROR\_MESSAGE );

}

} // end else

} // end method openFile

// read record from file

public void readRecord()

{

AccountRecord record;

// input the values from the file

try {

record = ( AccountRecord ) input.readObject(); // going from one type to the other type

// create array of Strings to display in GUI

String values[] = {

String.valueOf( record.getAccount() ),

record.getFirstName(),

record.getLastName()

};

// display record contents

userInterface.setFieldValues( values );

}

// display message when end-of-file reached

catch ( EOFException endOfFileException ) {

nextButton.setEnabled( false );

JOptionPane.showMessageDialog( this,

"No more records in file",

"End of File", JOptionPane.ERROR\_MESSAGE );

}

// display error message if cannot read object

// because class not found

catch ( ClassNotFoundException classNotFoundException ) {

JOptionPane.showMessageDialog( this,

"Unable to create object",

"Class Not Found", JOptionPane.ERROR\_MESSAGE );

}

// display error message if cannot read

// due to problem with file

catch ( IOException ioException ) {

JOptionPane.showMessageDialog( this,

"Error during read from file",

"Read Error", JOptionPane.ERROR\_MESSAGE );

}

}

// close file and terminate application

private void closeFile()

{

// close file and exit

try {

input.close();

System.exit( 0 );

}

// process exception while closing file

catch ( IOException ioException ) {

JOptionPane.showMessageDialog( this,

"Error closing file",

"Error", JOptionPane.ERROR\_MESSAGE );

System.exit( 1 );

}

}

// execute application; ReadSequentialFile constructor

// displays window

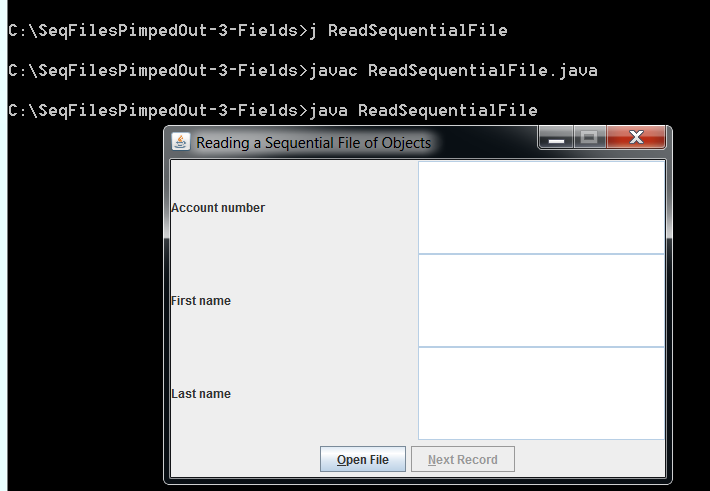
public static void main( String args[] )

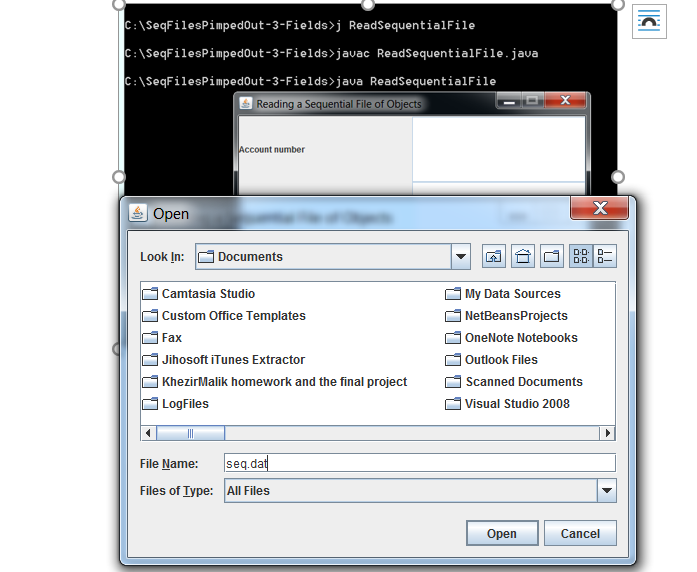
{

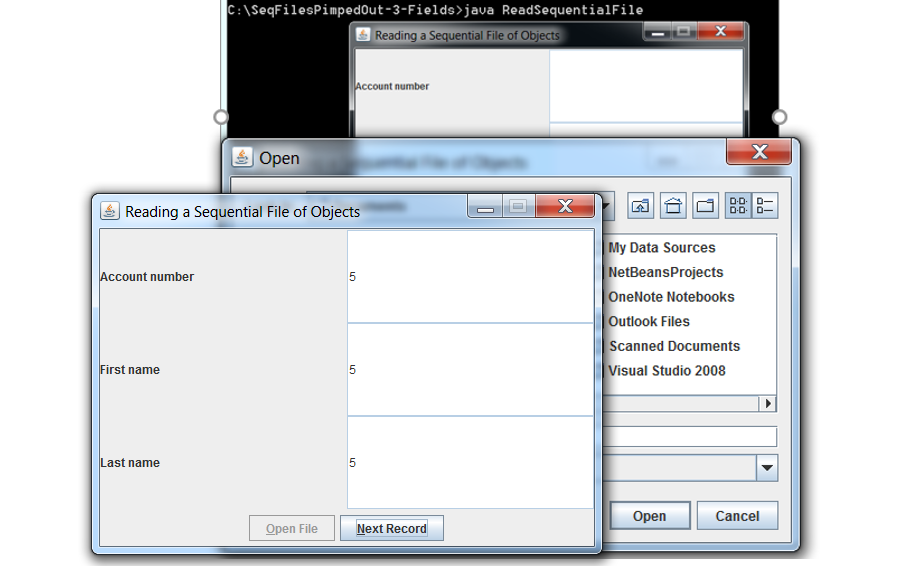
new ReadSequentialFile();

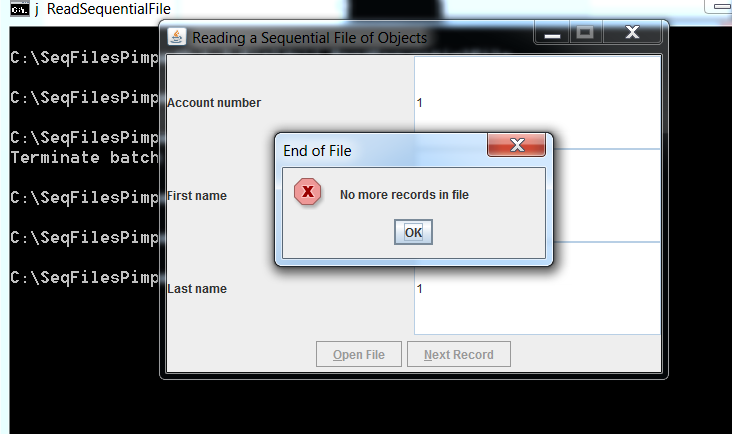
}

} // end class ReadSequentialFile









C’est Finis