Java Random BankUI 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, "", "", "", "", 0.0, 0.0,""); // default way of private variables.

}

// initialize a record

public AccountRecord(int acct, String first, String last, String addr, String soc, double bal,

double gpa,String title) {

setAccount(acct);

setFirstName(first);

setLastName(last);

setAddress(addr);

setSocSec(soc);

setBalance(bal);

setGPA(gpa);

setTitle(title);

}

// 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;

}

// set address

public void setAddress(String addr) {

address = addr;

}

// get address

public String getAddress() {

return address;

}

// set socsec

public void setSocSec(String soc) {

socsec = soc;

}

// get socsec

public String getSocSec() {

return socsec;

}

// set balance

public void setBalance(double bal) {

balance = bal;

}

// get balance

public double getBalance() {

return balance;

}

public void setGPA(double GPA) {

gpa = GPA;

}

public double getGPA() {

return gpa;

}

// get first name

public String getTitle() {

return this.title;

}

// set last name

public void setTitle(String title) {

this.title = title;

}

} // end class AccountRecord

//RandomAccessAccountRecord.java

// RandomAccessAccountRecord.java

// Subclass of AccountRecord for random access file programs.

// Java core packages

import java.io.\*;

public class RandomAccessAccountRecord extends AccountRecord {

// no-argument constructor calls other constructor

// with default values

public RandomAccessAccountRecord()

{

this( 0, "", "", 0.0 );

}

// initialize a RandomAccessAccountRecord

public RandomAccessAccountRecord( int account,

String firstName, String lastName, double balance )

{

super( account, firstName, lastName, balance );

}

// read a record from specified RandomAccessFile

public void read( RandomAccessFile file ) throws IOException

{

setAccount( file.readInt() );

setFirstName( padName( file,15 ));

setLastName( padName( file,28 ));

setBalance( file.readDouble() );

}

// ensure that name is proper length

private String padName( RandomAccessFile file, int ln )

throws IOException

{

char name[] = new char[ ln ], temp;

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

temp = file.readChar();

name[ count ] = temp;

}

return new String( name ).replace( '\0', ' ' );

}

// write a record to specified RandomAccessFile

public void write( RandomAccessFile file ) throws IOException

{

file.writeInt( getAccount() );

writeName( file, getFirstName(),15 );

writeName( file, getLastName(),28 );

file.writeDouble( getBalance() );

}

// write a name to file; maximum of 15 characters

private void writeName( RandomAccessFile file, String name,int ln)

throws IOException

{

StringBuffer buffer = null;

if ( name != null )

buffer = new StringBuffer( name );

else

buffer = new StringBuffer( ln);

buffer.setLength( ln );

file.writeChars( buffer.toString() );

}

// NOTE: This method contains a hard coded value for the

// size of a record of information.

public static int size()

{

return (30+4+56+8);

}

} // end class RandomAccessAccountRecord

// 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(8); // seven textfields

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],fieldValues[BankUI.ADDRESS],fieldValues[BankUI.SOCSEC], Double

.parseDouble(fieldValues[BankUI.BALANCE]),

Double.parseDouble(fieldValues[BankUI.GPA]),fieldValues[BankUI.TITLE]);

// 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

//CreateRandomFile.java

// Java core packages

import java.io.\*;

import java.awt.\*;

import java.awt.event.\*;

// Java extension packages

import javax.swing.\*;

import java.io.Serializable;

public class CreateRandomFile extends JFrame implements Serializable {

RandomAccessAccountRecord blankRecord;

private BankUI userInterface;

private JButton enterButton, openButton;

private static final long serialVersionUID = 7526471155622776147L;

// set up GUI

public CreateRandomFile()

{

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

// create instance of reusable user interface

userInterface = new BankUI( 4 ); // four textfields

getContentPane().add(

userInterface, BorderLayout.CENTER );

// 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);

// 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 ..." );

// 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

// 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 )

{

}

} // end anonymous inner class

); // end call to addActionListener

// register window listener to handle window closing event

addWindowListener(

// anonymous inner class to handle windowClosing event

new WindowAdapter() {

// add current record in GUI to file, then close file

public void windowClosing( WindowEvent event )

{

System.exit(0);

}

} // end anonymous inner class

); // end call to addWindowListener

setSize( 300, 200 );

// show();

setVisible(true);

} // end CreateRandomFile 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 == -1 )

return;

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

// return;

// get selected file

File fileName = fileChooser.getSelectedFile();

// 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 {

RandomAccessFile file =

new RandomAccessFile( fileName, "rw" );

RandomAccessAccountRecord blankRecord =

new RandomAccessAccountRecord();

// write 100 blank records

for ( int count = 0; count < 100; count++ )

blankRecord.write( file );

// close file

file.close();

// display message that file was created

JOptionPane.showMessageDialog( null,

"Created file " + fileName, "Status",

JOptionPane.INFORMATION\_MESSAGE );

System.exit( 0 ); // terminate program

}

// process exceptions during open, write or

// close file operations

catch ( IOException ioException ) {

JOptionPane.showMessageDialog( null,

"Error processing file", "Error processing file",

JOptionPane.ERROR\_MESSAGE );

System.exit( 1 );

}

}

} // end method openFile

// execute application; CreateRandomFile constructor

// displays window

public static void main( String args[] )

{

new CreateRandomFile();

System.out.printf("\n the value is %d ", JFileChooser.FILES\_ONLY);

}

} // end class CreateRandomFile

//READ ON ABOUT SERIALIZABLE AND UID IN SERIALIZABLE.TXT IN THIS DIRECTORY

//WriteRandomFile.java

// This program uses textfields to get information from the

// user at the keyboard and writes the information to a

// random-access file.

// Java core packages

import java.awt.\*;

import java.awt.event.\*;

import java.io.\*;

// Java extension packages

import javax.swing.\*;

public class WriteRandomFile extends JFrame {

private RandomAccessFile output;

private BankUI userInterface;

private JButton enterButton, openButton;

private JTextField setFcs;

// set up GUI

public WriteRandomFile()

{

super( "Write to random access file" );

// create instance of reusable user interface BankUI

userInterface = new BankUI( 4 ); // four textfields

getContentPane().add( userInterface,

BorderLayout.CENTER );

// get reference to generic task button doTask1 in BankUI

openButton = userInterface.getDoTask1Button();

openButton.setText( "Open..." );

// register listener to call openFile when button pressed

openButton.addActionListener(

// anonymous inner class to handle openButton event

new ActionListener() {

// allow user to select file to open

public void actionPerformed( ActionEvent event )

{

openFile();

}

} // end anonymous inner class

); // end call to addActionListener

// configure window

setDefaultCloseOperation(

WindowConstants.EXIT\_ON\_CLOSE );

// get reference to generic task button doTask2 in BankUI

enterButton = userInterface.getDoTask2Button();

enterButton.setText( "Enter" );

enterButton.setEnabled( false );

// register listener to call addRecord when button pressed

enterButton.addActionListener(

// anonymous inner class to handle enterButton event

new ActionListener() {

// add record to file

public void actionPerformed( ActionEvent event )

{

addRecord();

}

} // 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 \*/);

setSize( 300, 150 );

// show();

setVisible(true);

}

// enable user to choose file to open

private void openFile()

{

// display file dialog so user can select file

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 {

output = new RandomAccessFile( fileName, "rw" );

enterButton.setEnabled( true );

openButton.setEnabled( false );

}

// process exception while opening file

catch ( IOException ioException ) {

JOptionPane.showMessageDialog( this,

"File does not exist",

"Invalid File Name",

JOptionPane.ERROR\_MESSAGE );

}

}

} // end method openFile

// close file and terminate application

private void closeFile()

{

// close file and exit

try {

if ( output != null )

output.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 );

}

}

// add one record to file

public void addRecord()

{

int accountNumber = 0;

String fields[] = userInterface.getFieldValues();

RandomAccessAccountRecord record =

new RandomAccessAccountRecord();

// ensure account field has a value

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

// output values to file

try {

accountNumber =

Integer.parseInt( fields[ BankUI.ACCOUNT ] );

if ( accountNumber > 0 && accountNumber <= 100 ) {

record.setAccount( accountNumber );

record.setFirstName( fields[ BankUI.FIRSTNAME ] );

record.setLastName( fields[ BankUI.LASTNAME ] );

record.setBalance( Double.parseDouble(fields[ BankUI.BALANCE ] ) );

output.seek((long) ( accountNumber - 1 ) \*

RandomAccessAccountRecord.size() );

record.write( output );

}

userInterface.clearFields(); // clear TextFields

}

// process improper account number or balance format

catch ( NumberFormatException formatException ) {

JOptionPane.showMessageDialog( this,

"Bad account number or balance",

"Invalid Number Format",

JOptionPane.ERROR\_MESSAGE );

}

// process exceptions while writing to file

catch ( IOException ioException ) {

closeFile();

}

}

} // end method addRecord

// execute application

public static void main( String args[] )

{

new WriteRandomFile();

}

} // end class WriteRandomFile

//ReadRandomFile.java

// This program reads a random-access file sequentially and

// displays the contents one record at a time in text fields.

// Java core packages

import java.awt.\*;

import java.awt.event.\*;

import java.io.\*;

import java.text.DecimalFormat;

// Java extension packages

import javax.swing.\*;

public class ReadRandomFile extends JFrame {

private BankUI userInterface;

private RandomAccessFile input;

private JButton nextButton, openButton;

// set up GUI

public ReadRandomFile()

{

super( "Read Client File" );

// create reusable user interface instance

userInterface = new BankUI(4); // four textfields

getContentPane().add( userInterface );

// configure generic doTask1 button from BankUI

openButton = userInterface.getDoTask1Button();

openButton.setText( "Open File for Reading..." );

// register listener to call openFile when button pressed

openButton.addActionListener(

// anonymous inner class to handle openButton event

new ActionListener() {

// enable user to select file to open

public void actionPerformed( ActionEvent event )

{

openFile();

}

} // end anonymous inner class

); // end call to addActionListener

// configure generic doTask2 button from BankUI

nextButton = userInterface.getDoTask2Button();

nextButton.setText( "Next" );

nextButton.setEnabled( false );

// register listener to call readRecord when button pressed

nextButton.addActionListener(

// anonymous inner class to handle nextButton event

new ActionListener() {

// read a record when user clicks nextButton

public void actionPerformed( ActionEvent event )

{

readRecord();

}

} // end anonymous inner class

); // end call to addActionListener

// register listener for window closing event

addWindowListener(

// anonymous inner class to handle windowClosing event

new WindowAdapter() {

// close file and terminate application

public void windowClosing( WindowEvent event )

{

closeFile();

}

} // end anonymous inner class

); // end call to addWindowListener

setSize( 300, 150 );

// show();

setVisible(true);

}

// enable user to select file to open

private void openFile()

{

// display file dialog so user can select file

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 is 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 RandomAccessFile( fileName, "r" );

nextButton.setEnabled( true );

openButton.setEnabled( false );

}

// catch exception while opening file

catch ( IOException ioException ) {

JOptionPane.showMessageDialog( this,

"File does not exist", "Invalid File Name",

JOptionPane.ERROR\_MESSAGE );

}

}

} // end method openFile

// read one record

public void readRecord()

{

DecimalFormat twoDigits = new DecimalFormat( "0.00" );

RandomAccessAccountRecord record =

new RandomAccessAccountRecord();

// read a record and display

try {

//Skip blank records in the file

do {

record.read( input );

} while ( record.getAccount() == 0 );

String values[] = {

String.valueOf( record.getAccount() ),

record.getFirstName(),

record.getLastName(),

String.valueOf( record.getBalance() )

};

userInterface.setFieldValues( values );

}

// close file when end-of-file reached

catch ( EOFException eofException ) {

JOptionPane.showMessageDialog( this, "No more records",

"End-of-file reached",

JOptionPane.INFORMATION\_MESSAGE );

closeFile();

}

// process exceptions from problem with file

catch ( IOException ioException ) {

JOptionPane.showMessageDialog( this,

"Error Reading File", "Error",

JOptionPane.ERROR\_MESSAGE );

System.exit( 1 );

}

} // end method readRecord

// close file and terminate application

private void closeFile()

{

// close file and exit

try {

if ( input != null )

input.close();

System.exit( 0 );

}

// process exception closing file

catch( IOException ioException ) {

JOptionPane.showMessageDialog( this,

"Error closing file",

"Error", JOptionPane.ERROR\_MESSAGE );

System.exit( 1 );

}

}

// execute application

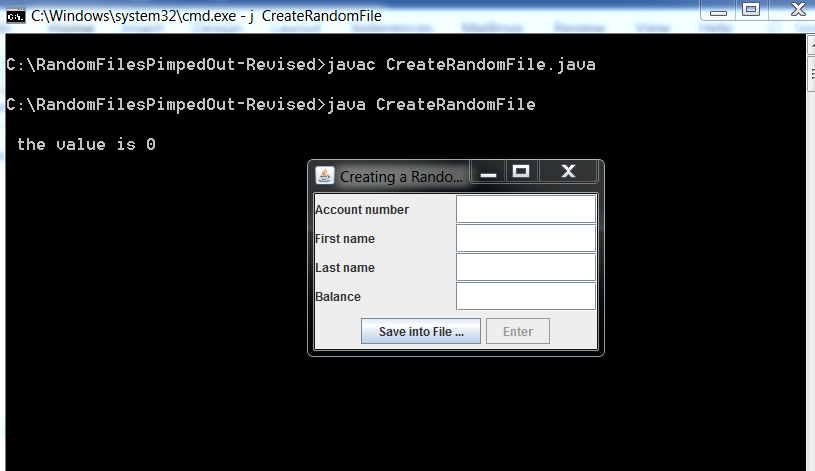
public static void main( String args[] )

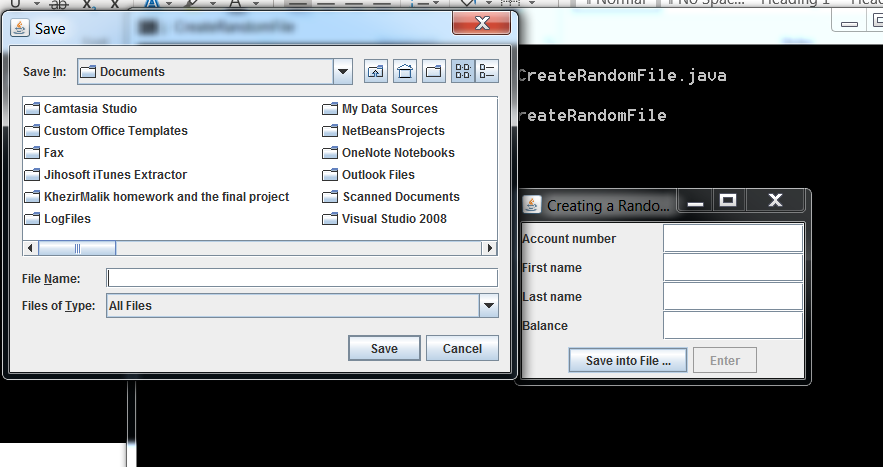
{

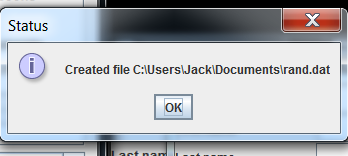
new ReadRandomFile();

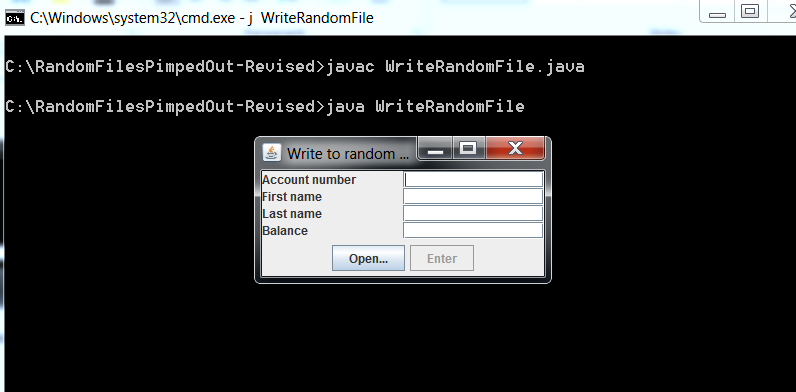
}

} // end class ReadRandomFile

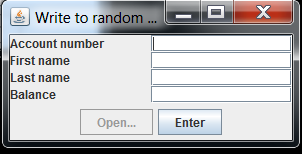




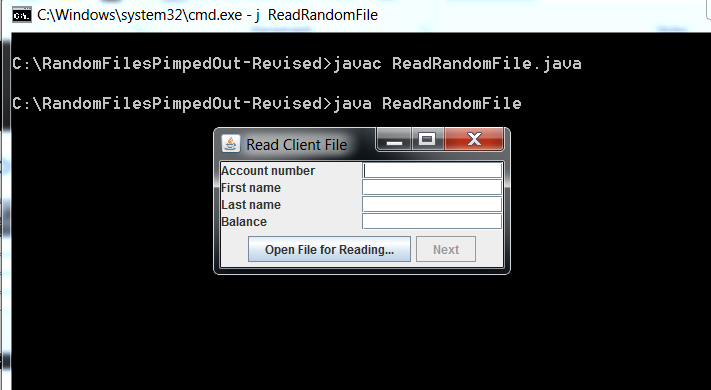




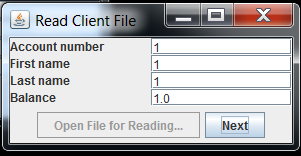
Click on the OPEN… button Type rand.dat an see…..

Notice the OPEN… button is grayed out and the Enter button is now active!

Add data!!!!! Use numbers it’s easier.



Open rand.dat and then click the Next button.

Once again the Open File for Reading button is grayed out and the

Next button is active!!!!!!!

C’est Finis