

# Stage C

## Source code

```
// ReadRandomFile.java
// This program reads a random-access file randomly
// 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.*;

// Deitel packages (referenced from Deitel textbook)
import Record.Data;
import Record.RandomAccessData;

import Record.*;

public class ReadRandomFile extends JFrame {
    private Data userInterface;
    private RandomAccessFile input;
    private JButton nextButton, openButton;

    // GUI setup
    public ReadRandomFile()
    {
        super( "Read Client File" );

        // create reusable user interface instance
        userInterface = new Data( 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( 400, 500 );
show();
}

// 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" );
    RandomAccessData record =
        new RandomAccessData();

    // read a record and display
    try {

```

```

do {
    record.read( input );
} while ( record.getAccount() == 0 );

String values[] = {
    String.valueOf( record.getAccount() ),
    record.getFirstName(),
    String.valueOf( record.getQuantity() ),
    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

import java.awt.*;
import java.awt.event.*;
import java.io.*;

// Java extension packages
import javax.swing.*;

// Deitel packages (referenced from the Deitel textbook)
import Record.*;

public class WriteRandomFile extends JFrame {
    private RandomAccessFile output;
    private Data userInterface;
    private JButton enterButton, openButton;

    // set up GUI
    public WriteRandomFile()
    {
        super( "Write to random access file" );

        // create instance of reusable user interface EmailStorage
        userInterface = new Data( ); // number of textfields created
        getContentPane().add( userInterface,
            BorderLayout.CENTER );

        // get reference to generic task button doTask1 in EmailStorage
        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

// register window listener for window closing event
addWindowListener(

    // anonymous inner class to handle windowClosing event
    new WindowAdapter() {

        // add record in GUI, then close file
        public void windowClosing( WindowEvent event )
        {
            if ( output != null )
                addRecord();

            closeFile();
        }

    } // end anonymous inner class

); // end call to addWindowListener

// get reference to generic task button doTask2 in EmailStorage
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

setSize( 400, 500 );
show();
}

// enable user to choose file to open
private void openFile()
{
    // display file dialog so user can select file

```

```

JFileChooser fileChooser = new JFileChooser();
fileChooser.setSelectionMode(
    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();
    RandomAccessData record =
        new RandomAccessData();

    // ensure account field has a value
    if ( ! fields[ Data.ACCOUNT ].equals( "" ) ) {

        // output values to file
        try {
            accountRecords =
                Integer.parseInt( fields[ Data.ACCOUNT ] );

            if ( accountRecords > 0 && accountRecords <= 100 ) {
                record.setAccount( accountRecords);

                record.setName( fields[ Data.MAILTO ] );
                record.setDate( Double.parseDouble(
                    fields[ Data.DATE ] ) );
                record.setName2( Double.parseDouble(
                    fields[ Data.MAILFROM ] ) );

                output.seek( ( accountRecords - 1 ) *
                    RandomAccessData.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

// Fig. 16.20: TransactionProcessor.java
// Transaction processing program using RandomAccessFiles.
// This program reads a random-access file sequentially,
// updates records already written to the file, creates new
// records to be placed in the file and deletes data
// already in the file.

// Java core packages
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.text.DecimalFormat;

// Java extension packages
import javax.swing.*;

// Programmer Developed Packages
import Record.*;

public class Processor extends JFrame {
    private UpdateDialog updateDialog;
    private NewDialog newDialog;
    private DeleteDialog deleteDialog;
    private JMenuItem newItem, updateItem, deleteItem,
        openItem, exitItem;
    private JDesktopPane desktop;
    private RandomAccessFile file;
    private RandomAccessData record;

    // set up GUI
    public TransactionProcessor()
    {
        super( "Transaction Processor" );

        // set up desktop, menu bar and File menu

```

```

desktop = new JDesktopPane();
getContentPane().add( desktop );

JMenuBar menuBar = new JMenuBar();
setJMenuBar( menuBar );

JMenu fileMenu = new JMenu( "File" );
menuBar.add( fileMenu );

// set up menu item for adding a record
newItem = new JMenuItem( "New Record" );
newItem.setEnabled( false );

// display new record dialog when user selects New Record
newItem.addActionListener(

    new ActionListener() {

        public void actionPerformed((ActionEvent event)
        {
            newDialog.setVisible( true );
        }
    }
);

// set up menu item for updating a record
updateItem = new JMenuItem( "Update Record" );
updateItem.setEnabled( false );

// display update dialog when user selects Update Record
updateItem.addActionListener(

    new ActionListener() {

        public void actionPerformed((ActionEvent event)
        {
            updateDialog.setVisible( true );
        }
    }
);

// set up menu item for deleting a record
deleteItem = new JMenuItem( "Delete Record" );
deleteItem.setEnabled( false );

// display delete dialog when user selects Delete Record
deleteItem.addActionListener(

    new ActionListener() {

        public void actionPerformed((ActionEvent event)

```

```

        {
            deleteDialog.setVisible( true );
        }
    }
);

// set up button for opening file
openItem = new JMenuItem( "New/Open File" );

// enable user to select file to open, then set up
// dialog boxes
openItem.addActionListener(

    new ActionListener() {

        public void actionPerformed( ActionEvent event )
        {
            boolean opened = openFile();

            if ( !opened )
                return;

            openItem.setEnabled( false );

            // set up internal frames for record processing
            updateDialog = new UpdateDialog( file );
            desktop.add( updateDialog );

            deleteDialog = new DeleteDialog( file );
            desktop.add ( deleteDialog );

            newDialog = new NewDialog( file );
            desktop.add( newDialog );
        }

    } // end anonymous inner class

); // end call to addActionListener

// set up menu item for exiting program
exitItem = new JMenuItem( "Exit" );
exitItem.setEnabled( true );

// terminate application
exitItem.addActionListener(

    new ActionListener() {

        public void actionPerformed( ActionEvent event )
        {
            closeFile();
        }
    }
);

```

```

    }
  }
);

// attach menu items to File menu
fileMenu.add( openItem );
fileMenu.add( newItem );
fileMenu.add( updateItem );
fileMenu.add( deleteItem );
fileMenu.addSeparator();
fileMenu.add( exitItem );

// configure window
setDefaultCloseOperation(
  WindowConstants.DO_NOTHING_ON_CLOSE );

setSize( 400, 250 );
setVisible( true );

} // end TransactionProcessor constructor

// enable user to select file to open
private boolean openFile()
{
  // display 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 false;

  // 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 );

    return false;
  }

  else {

    // open file

```

```

try {
    file = new RandomAccessFile( fileName, "rw" );
    openItem.setEnabled( false );
    newItem.setEnabled( true );
    updateItem.setEnabled( true );
    deleteItem.setEnabled( true );
}

// process problems opening file
catch ( IOException ioException ) {
    JOptionPane.showMessageDialog( this,
        "File does not exist", "Invalid File Name",
        JOptionPane.ERROR_MESSAGE );

    return false;
}

return true; // file opened
}

// close file and terminate application
private void closeFile()
{
    // close file and exit
    try {
        if ( file != null )
            file.close();

        System.exit( 0 );
    }

    // process exceptions 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 Processor();
}

} // end class Processor

// Java core packages
import java.awt.*;

```

```

import java.awt.event.*;
import java.io.*;
import java.text.DecimalFormat;

// Java extension packages
import javax.swing.*;

// Programmer Developed Packages
import Record.*;

// class for creating new records
class NewDialog extends JFrame {
    private RandomAccessFile file;
    private Data userInterface;

    // set up GUI
    public NewDialog( RandomAccessFile newFile )
    {
        super( "New Record" );

        file = newFile;

        // attach user interface to dialog
        userInterface = new Data( 4 );
        getContentPane().add( userInterface,
            BorderLayout.CENTER );

        // set up Save Changes button and register listener
        JButton saveButton = userInterface.getDoTask1Button();
        saveButton.setText( "Save Changes" );

        saveButton.addActionListener(

            new ActionListener() {

                // add new record to file
                public void actionPerformed( ActionEvent event )
                {
                    addRecord( getRecord() );
                    setVisible( false );
                    userInterface.clearFields();
                }

            } // end anonymous inner class

        ); // end call to addActionListener

        JButton cancelButton = userInterface.getDoTask2Button();
        cancelButton.setText( "Cancel" );

        cancelButton.addActionListener(

```

```

new ActionListener() {

    // dismiss dialog without storing new record
    public void actionPerformed( ActionEvent event )
    {
        setVisible( false );
        userInterface.clearFields();
    }

} // end anonymous inner class

); // end call to addActionListener

setSize( 300, 150 );
setVisible( false );

} // end constructor

// get record from file
private RandomAccessData getRecord()
{
    RandomAccessData record =
        new RandomAccessData();

    // get record from file
    try {
        JTextField accountField =
            userInterface.getFields()[ Data.ACCOUNT ];

        int accountNumber =
            Integer.parseInt( accountField.getText() );

        if ( accountNumber < 1 || accountNumber > 100 ) {
            JOptionPane.showMessageDialog( this,
                "Account Does Not Exist",
                "Error", JOptionPane.ERROR_MESSAGE );
            return record;
        }

        // seek to record location
        file.seek( ( accountNumber - 1 ) *
            RandomAccessData.size() );

        // read record from file
        record.read( file );
    }

    // process invalid account number format
    catch ( NumberFormatException numberFormat ) {
        JOptionPane.showMessageDialog( this,

```

```

        "Account Does Not Exist", "Invalid Number Format",
        JOptionPane.ERROR_MESSAGE );
    }

    // process file processing problems
    catch ( IOException ioException ) {
        JOptionPane.showMessageDialog( this,
            "Error Reading File",
            "Error", JOptionPane.ERROR_MESSAGE );
    }

    return record;

} // end method getRecord

// add record to file
public void addRecord( RandomAccessData record )
{
    String[] fields = userInterface.getFieldValues();

    if ( record.getAccount() != 0 ) {
        JOptionPane.showMessageDialog( this,
            "Record Already Exists",
            "Error", JOptionPane.ERROR_MESSAGE );
        return;
    }

    // output the values to the file
    try {

        // set account, first name, last name and balance
        // for record
        record.setAccount( Integer.parseInt(
            fields[ Data.ACCOUNT ] ) );
        record.setFirstName( fields[ Data.FIRSTNAME ] );
        record.setLastName( fields[ Data.LASTNAME ] );
        record.setBalance( Double.parseDouble(
            fields[ Data.BALANCE ] ) );

        // seek to record location
        file.seek( ( record.getAccount() - 1 ) *
            RandomAccessData.size() );

        // write record
        record.write( file );
    }

    // process invalid account or balance format
    catch ( NumberFormatException numberFormat ) {
        JOptionPane.showMessageDialog( this,
            "Invalid Balance", "Invalid Number Format",

```

```

        JOptionPane.ERROR_MESSAGE );
    }

    // process file processing problems
    catch ( IOException ioException ) {
        JOptionPane.showMessageDialog( this,
            "Error Writing To File",
            "Error", JOptionPane.ERROR_MESSAGE );
    }

} // end method addRecord

} // end class NewDialog

// Java core packages
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.text.DecimalFormat;

// Java extension packages
import javax.swing.*;

// Programmer Developed Packages
import Record.*;

// class for deleting records
class DeleteDialog extends JFrame {
    private RandomAccessFile file; // file for output
    private Data userInterface;

    // set up GUI
    public DeleteDialog( RandomAccessFile deleteFile )
    {
        super( "Delete Record" );

        file = deleteFile;

        // create BankUI with only account field
        userInterface = new Data( 1 );

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

        // set up Delete Record button and register listener
        JButton deleteButton = userInterface.getDoTask1Button();
        deleteButton.setText( "Delete Record" );

        deleteButton.addActionListener(

            new ActionListener() {

```

```

// overwrite existing record
public void actionPerformed((ActionEvent event)
{
    addRecord( getRecord() );
    setVisible( false );
    userInterface.clearFields();
}

} // end anonymous inner class

); // end call to addActionListener

// set up Cancel button and register listener
JButton cancelButton = userInterface.getDoTask2Button();
cancelButton.setText( "Cancel" );

cancelButton.addActionListener(

    new ActionListener() {

        // cancel delete operation by hiding dialog
        public void actionPerformed( ActionEvent event )
        {
            setVisible( false );
        }

    } // end anonymous inner class

); // end call to addActionListener

// set up listener for account text field
JTextField accountField =
    userInterface.getFields()[ Data.ACCOUNT ];

accountField.addActionListener(

    new ActionListener() {

        public void actionPerformed( ActionEvent event )
        {
            RandomAccessData record = getRecord();
        }

    } // end anonymous inner class

); // end call to addActionListener

setSize( 300, 100 );
setVisible( false );

```

```

} // end constructor

// get record from file
private RandomAccessData getRecord()
{
    RandomAccessData record =
        new RandomAccessData();

    // get record from file
    try {
        JTextField accountField =
            userInterface.getFields()[ Data.ACCOUNT ];

        int accountNumber =
            Integer.parseInt( accountField.getText() );

        if ( accountNumber < 1 || accountNumber > 100 ) {
            JOptionPane.showMessageDialog( this,
                "Account Does Not Exist",
                "Error", JOptionPane.ERROR_MESSAGE );
            return( record );
        }

        // seek to record location and read record
        file.seek( ( accountNumber - 1 ) *
            RandomAccessData.size() );
        record.read( file );

        if ( record.getAccount() == 0 )
            JOptionPane.showMessageDialog( this,
                "Account Does Not Exist",
                "Error", JOptionPane.ERROR_MESSAGE );
    }

    // process invalid account number format
    catch ( NumberFormatException numberFormat ) {
        JOptionPane.showMessageDialog( this,
            "Account Does Not Exist",
            "Invalid Number Format",
            JOptionPane.ERROR_MESSAGE );
    }

    // process file processing problems
    catch ( IOException ioException ) {
        JOptionPane.showMessageDialog( this,
            "Error Reading File",
            "Error", JOptionPane.ERROR_MESSAGE );
    }

    return record;
}

```

```

} // end method getRecord

// add record to file
public void addRecord( RandomAccessData record )
{
    if ( record.getAccount() == 0 )
        return;

    // delete record by setting account number to 0
    try {
        int accountNumber = record.getAccount();

        // seek to record position
        file.seek( ( accountNumber - 1 ) *
            RandomAccessData.size() );

        // set account to 0 and overwrite record
        record.setAccount( 0 );
        record.write( file );
    }

    // process file processing problems
    catch ( IOException ioException ) {
        JOptionPane.showMessageDialog( this,
            "Error Writing To File",
            "Error", JOptionPane.ERROR_MESSAGE );
    }

} // end method addRecord

} // end class DeleteDialog

// Java core packages
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.text.DecimalFormat;

// Java extension packages
import javax.swing.*;

// Programmer Developed Packages
import Record.*;

// class for updating records
class UpdateDialog extends JFrame
{
    private RandomAccessFile file;
    private Data userInterface;

    // set up GUI

```

```

public UpdateDialog( RandomAccessFile updateFile )
{
    super( "Update Record" );

    file = updateFile;

    // set up GUI components
    userInterface = new Data( 5 );
    getContentPane().add( userInterface,
        BorderLayout.CENTER );

    // set up Save Changes button and register listener
    JButton saveButton = userInterface.getDoTask1Button();
    saveButton.setText( "Save Changes" );

    saveButton.addActionListener(

        new ActionListener() {

            public void actionPerformed( ActionEvent event )
            {
                addRecord( getRecord() );
                setVisible( false );
                userInterface.clearFields();
            }
        }
    );

    // set up Cancel button and register listener
    JButton cancelButton = userInterface.getDoTask2Button();
    cancelButton.setText( "Cancel" );

    cancelButton.addActionListener(

        new ActionListener() {

            public void actionPerformed( ActionEvent event )
            {
                setVisible( false );
                userInterface.clearFields();
            }
        }
    );

    // set up listener for transaction textfield
    JTextField transactionField =
        userInterface.getFields()[ Data.TRANSACTION ];

    transactionField.addActionListener(

        new ActionListener() {

```

```

public void actionPerformed((ActionEvent event)
{
// add transaction amount to balance
try {
    RandomAccessData record = getRecord();

    // get textfield values from userInterface
    String fieldValues[] =
        userInterface.getFieldValues();

    // get transaction amount
    double change = Double.parseDouble(
        fieldValues[ Data.TRANSACTION ] );

    // specify Strings to display in GUI
    String[] values = {
        String.valueOf( record.getAccount() ),
        record.getMailTo(),
        record.getMailFrom(),
        record.getContent(),
        String.valueOf( record.getDate ()
            + change ),
        "Charge(+) or payment (-)" };

    // display Strings in GUI
    userInterface.setFieldValues( values );
}

// process invalid number in transaction field
catch ( NumberFormatException numberFormat ) {
    JOptionPane.showMessageDialog( null,
        "Invalid Transaction",
        "Invalid Number Format",
        JOptionPane.ERROR_MESSAGE );
}

} // end method actionPerformed

} // end anonymous inner class

); // end call to addActionListener

// set up listener for account text field
JTextField accountField =
    userInterface.getFields()[ Data.ACCOUNT ];

accountField.addActionListener(

    new ActionListener() {

```

```

// get record and display contents in GUI
public void actionPerformed( ActionEvent event )
{
    RandomAccessData record = getRecord();

    if ( record.getAccount() != 0 ) {
        String values[] = {
            String.valueOf( record.getAccount() ),
            record.getFirstName(),
            record.getLastName(),
            String.valueOf( record.getBalance() ),
            "Charge(+) or payment (-)" };

        userInterface.setFieldValues( values );
    }

} // end method actionPerformed

} // end anonymous inner class

); // end call to addActionListener

setSize( 300, 175 );
setVisible( false );
}

// get record from file
private RandomAccessData getRecord()
{
    RandomAccessData record =
        new RandomAccessData();

    // get record from file
    try {
        JTextField accountField =
            userInterface.getFields()[ Data.ACCOUNT ];

        int accountNumber =
            Integer.parseInt( accountField.getText() );

        if ( accountNumber < 1 || accountNumber > 100 ) {
            JOptionPane.showMessageDialog( this,
                "Account Does Not Exist",
                "Error", JOptionPane.ERROR_MESSAGE );
            return record;
        }

        // seek to appropriate record location in file
        file.seek( ( accountNumber - 1 ) *
            RandomAccessData.size() );
        record.read( file );
    }
}

```

```

    if ( record.getAccount() == 0 )
        JOptionPane.showMessageDialog( this,
            "Account Does Not Exist",
            "Error", JOptionPane.ERROR_MESSAGE );
    }

    // process invalid account number format
    catch ( NumberFormatException numberFormat ) {
        JOptionPane.showMessageDialog( this,
            "Invalid Account", "Invalid Number Format",
            JOptionPane.ERROR_MESSAGE );
    }

    // process file processing problems
    catch ( IOException ioException ) {
        JOptionPane.showMessageDialog( this,
            "Error Reading File",
            "Error", JOptionPane.ERROR_MESSAGE );
    }

    return record;
} // end method getRecord

// add record to file
public void addRecord( RandomAccessData record )
{
    // update record in file
    try {
        int accountNumber = record.getAccount();

        file.seek( ( accountNumber - 1 ) *
            RandomAccessData.size() );

        String[] values = userInterface.getFieldValues();

        // set firstName, lastName and balance in record
        record.setFirstName( values[ Data.FIRSTNAME ] );
        record.setLastName( values[ Data.LASTNAME ] );
        record.setBalance(
            Double.parseDouble( values[ Data.BALANCE ] ) );

        // rewrite record to file
        record.write( file );
    }

    // process file processing problems
    catch ( IOException ioException ) {
        JOptionPane.showMessageDialog( this,
            "Error Writing To File",

```

```

        "Error", JOptionPane.ERROR_MESSAGE );
    }

    // process invalid balance value
    catch ( NumberFormatException numberFormat ) {
        JOptionPane.showMessageDialog( this,
            "Bad Balance", "Invalid Number Format",
            JOptionPane.ERROR_MESSAGE );
    }

} // end method addRecord

} // end class UpdateDialog

public class Search
{

    Analyzer analyzer = new StandardAnalyzer();

    // Store the index in memory:
    Directory directory = new RAMDirectory();
    // To store an index on disk, use this instead:
    //Directory directory = FSDirectory.getDirectory("/tmp/testindex");
    IndexWriter iwriter = new IndexWriter(directory, analyzer, true);
    iwriter.setMaxFieldLength(25000);
    Document doc = new Document();
    String text = "This is the text to be indexed.";
    doc.add(new Field("fieldname", text, Field.Store.YES,
        Field.Index.TOKENIZED));
    iwriter.addDocument(doc);
    iwriter.optimize();
    iwriter.close();

    // Now search the index:
    IndexSearcher isearcher = new IndexSearcher(directory);
    // Parse a simple query that searches for "text":
    QueryParser parser = new QueryParser("fieldname", analyzer);
    Query query = parser.parse("text");
    Hits hits = isearcher.search(query);
    assertEquals(1, hits.length());
    // Iterate through the results:
    for (int i = 0; i < hits.length(); i++) {
        Document hitDoc = hits.doc(i);
        assertEquals("This is the text to be indexed.", hitDoc.get("fieldname"));
    }
    isearcher.close();
    directory.close();

}

```

```

package Record;

//AccountRecord

// Java core packages
import java.io.Serializable;

public class EmailStorage implements Serializable {
    private int account;
    private String mailto;
    private String date;
    private String mailfrom;
    private String content;

    // no-argument constructor calls other constructor with
    // default values
    public EmailStorage()
    {
        this( 0, "", "", "", "" );
    }

    // initialize a record
    public EmailStorage( int acct, String mailto,String date,String mailfrom,String content
    )
    {
        setAccount( acct );
        setAddress1( mailto);
        setDate( date );
        setAddress2( mailfrom);
        setContent( content );
    }

    // set account number
    public void setAccount( int acct )
    {
        account = acct;
    }

    // get account number
    public int getAccount()
    {
        return account;
    }

    // setAddress to send
    public void setAddress1( String first )
    {
        //write if input is zero return
        mailto = first;
    }
}

```

```

// get Adress to send
public String getAddress1()
{
    return mailto;
}

// set Date
public void setDate( String quant)
{
    date = quant;
}

public void setAddress2( String first )
{
    //write if input is zero return
    mailfrom = first;
}

// get Date
public String getAddress2()
{
    return mailfrom;
}

// get Adress from
public String getDate()
{
    return date;
}

// set address
public void setContent( String content )
{
    content = content;
}

// get Content
public String getContent()
{
    return content;
}

} // end class EmailStorage

package Record;
// Fig. 16.11: RandomAccessAccountRecord.java
// Subclass of AccountRecord for random access file programs.

// Java core packages

```

```

import java.io.*;

public class RandomAccessData extends EmailStorage {

    private String title;

    // no-argument constructor calls other constructor
    // with default values
    public RandomAccessData()
    {
        this( 0, "", "", "", "" );
    }

    // initialize a RandomAccessAccountRecord
    public RandomAccessData( int account,
        String mailto,String date,String mailfrom,String content)
    {
        super( account, mailto, date, mailfrom, content );
    }

    // read a record from specified RandomAccessFile
    public void read( RandomAccessFile file ) throws IOException
    {
        setAccount( file.readInt() );
        setAddress1(padName(file) );
        setDate(padName(file) );
        setAddress2(padName(file) );
        setContent(padName(file) );
    }
    /**
    static final int TITLE_BYTE_SIZE = 20;

    public void setTitle(String title)
    {
        StringBuffer buf = new StringBuffer(title);
        buf.setLength(TITLE_BYTE_SIZE);
        this.title = buf.toString();
    }
    */

    // ensure that name is proper length
    private String padName( RandomAccessFile file )
    throws IOException
    {
        char name[] = new char[ 15 ], 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, getAddress1() );
    writeName( file, getDate() );
    writeName( file, getAddress2() );
    writeName( file, getContent() );
}

// write a name to file; maximum of 15 characters
private void writeName( RandomAccessFile file, String name )
    throws IOException
{
    StringBuffer buffer = null;

    if ( name != null )
        buffer = new StringBuffer( name );
    else
        buffer = new StringBuffer( 15 );

    buffer.setLength( 15 );
    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 72;
}
*/

/**
 * @pre Size of record is not known.
 * @post Size of record is calculated as sum of lengths of fields.
 *
 */
public static int size()
{
    // FIELDS of RECORD
    return 4 // 4 bytes for primitive int: account number
        + 30 // 15 char (2 bytes each) array: first name
        + 4 // 15 char (2 bytes each) array: last name
        + 8 ; // 8 bytes for primitive double: account balance
}
// -----

```

```
// 72 ==> sum of bytes == length of this data record.  
// =====  
} // end method size()
```

```
} // end class RandomAccessData
```