Package homeinventory;

Import javax.swing.\*;

Import javax.swing.filechooser.\*;

Import java.awt.\*;

Import java.awt.event.\*;

Import java.beans.\*;

Import com.toedter.calendar.\*;

Import java.awt.geom.\*;

Import java.io.\*;

Import java.util.\*;

Import java.text.\*;

Import java.awt.print.\*;

Public class HomeInventory extends JFrame

{

// Toolbar

JToolBar inventoryToolBar = new JToolBar();

JButton newButton = new JButton(new ImageIcon(“new.gif”));

JButton deleteButton = new JButton(new ImageIcon(“delete.gif”));

JButton saveButton = new JButton(new ImageIcon(“save.gif”));

JButton previousButton = new JButton(new ImageIcon(“previous.gif”));

JButton nextButton = new JButton(new ImageIcon(“next.gif”));

JButton printButton = new JButton(new ImageIcon(“print.gif”));

JButton exitButton = new JButton();

// Frame

JLabel itemLabel = new JLabel();

JTextField itemTextField = new JTextField();

JLabel locationLabel = new JLabel();

JComboBox locationComboBox = new JComboBox();

JCheckBox markedCheckBox = new JCheckBox();

JLabel serialLabel = new JLabel();

JTextField serialTextField = new JTextField();

JLabel priceLabel = new JLabel();

JTextField priceTextField = new JTextField();

JLabel dateLabel = new JLabel();

JDateChooser dateDateChooser = new JDateChooser();

JLabel storeLabel = new JLabel();

JTextField storeTextField = new JTextField();

JLabel noteLabel = new JLabel();

JTextField noteTextField = new JTextField();

JLabel photoLabel = new JLabel();

Static JTextArea photoTextArea = new JTextArea();

JButton photoButton = new JButton();

JPanel searchPanel = new JPanel();

JButton[] searchButton = new JButton[26];

PhotoPanel photoPanel = new PhotoPanel();

Static final int maximumEntries = 300;

Static int numberEntries;

Static InventoryItem[] myInventory = new InventoryItem[maximumEntries];

Int currentEntry;

Static final int entriesPerPage = 2;

Static int lastPage;

Public static void main(String args[])

{

// create frame

New HomeInventory().show();

}

Public HomeInventory()

{

// frame constructor

setTitle(“Home Inventory Manager”);

setResizable(false);

setDefaultCloseOperation(JFrame.DO\_NOTHING\_ON\_CLOSE);

addWindowListener(new WindowAdapter()

{

Public void windowClosing(WindowEvent evt)

{

exitForm(evt);

}

});

getContentPane().setLayout(new GridBagLayout());

GridBagConstraints gridConstraints;

inventoryToolBar.setFloatable(false);

inventoryToolBar.setBackground(Color.BLUE);

inventoryToolBar.setOrientation(SwingConstants.VERTICAL);

gridConstraints = new GridBagConstraints();

gridConstraints.gridx = 0;

gridConstraints.gridy = 0;

gridConstraints.gridheight = 8;

gridConstraints.fill = GridBagConstraints.VERTICAL;

getContentPane().add(inventoryToolBar, gridConstraints);

inventoryToolBar.addSeparator();

Dimension bSize = new Dimension(70, 50);

newButton.setText(“New”);

sizeButton(newButton, bSize);

newButton.setToolTipText(“Add New Item”);

newButton.setHorizontalTextPosition(SwingConstants.CENTER);

newButton.setVerticalTextPosition(SwingConstants.BOTTOM);

newButton.setFocusable(false);

inventoryToolBar.add(newButton);

newButton.addActionListener(new ActionListener()

{

Public void actionPerformed(ActionEvent e)

{

newButtonActionPerformed€;

}

});

deleteButton.setText(“Delete”);

sizeButton(deleteButton, bSize);

deleteButton.setToolTipText(“Delete Current Item”);

deleteButton.setHorizontalTextPosition(SwingConstants.CENTER);

deleteButton.setVerticalTextPosition(SwingConstants.BOTTOM);

deleteButton.setFocusable(false);

inventoryToolBar.add(deleteButton);

deleteButton.addActionListener(new ActionListener()

{

Public void actionPerformed(ActionEvent e)

{

deleteButtonActionPerformed€;

}

});

saveButton.setText(“Save”);

sizeButton(saveButton, bSize);

saveButton.setToolTipText(“Save Current Item”);

saveButton.setHorizontalTextPosition(SwingConstants.CENTER);

saveButton.setVerticalTextPosition(SwingConstants.BOTTOM);

saveButton.setFocusable(false);

inventoryToolBar.add(saveButton);

saveButton.addActionListener(new ActionListener()

{

Public void actionPerformed(ActionEvent e)

{

saveButtonActionPerformed€;

}

});

inventoryToolBar.addSeparator();

previousButton.setText(“Previous”);

sizeButton(previousButton, bSize);

previousButton.setToolTipText(“Display Previous Item”);

previousButton.setHorizontalTextPosition(SwingConstants.CENTER);

previousButton.setVerticalTextPosition(SwingConstants.BOTTOM);

previousButton.setFocusable(false);

inventoryToolBar.add(previousButton);

previousButton.addActionListener(new ActionListener()

{

Public void actionPerformed(ActionEvent e)

{

previousButtonActionPerformed€;

}

});

nextButton.setText(“Next”);

sizeButton(nextButton, bSize);

nextButton.setToolTipText(“Display Next Item”);

nextButton.setHorizontalTextPosition(SwingConstants.CENTER);

nextButton.setVerticalTextPosition(SwingConstants.BOTTOM);

nextButton.setFocusable(false);

inventoryToolBar.add(nextButton);

nextButton.addActionListener(new ActionListener()

{

Public void actionPerformed(ActionEvent e)

{

nextButtonActionPerformed€;

}

});

inventoryToolBar.addSeparator();

printButton.setText(“Print”);

sizeButton(printButton, bSize);

printButton.setToolTipText(“Print Inventory List”);

printButton.setHorizontalTextPosition(SwingConstants.CENTER);

printButton.setVerticalTextPosition(SwingConstants.BOTTOM);

printButton.setFocusable(false);

inventoryToolBar.add(printButton);

printButton.addActionListener(new ActionListener()

{

Public void actionPerformed(ActionEvent e)

{

printButtonActionPerformed€;

}

});

exitButton.setText(“Exit”);

sizeButton(exitButton, bSize);

exitButton.setToolTipText(“Exit Program”);

exitButton.setFocusable(false);

inventoryToolBar.add(exitButton);

exitButton.addActionListener(new ActionListener()

{

Public void actionPerformed(ActionEvent e)

{

exitButtonActionPerformed€;

}

});

itemLabel.setText(“Inventory Item”);

gridConstraints = new GridBagConstraints();

gridConstraints.gridx = 1;

gridConstraints.gridy = 0;

gridConstraints.insets = new Insets(10, 10, 0, 10);

gridConstraints.anchor = GridBagConstraints.EAST;

getContentPane().add(itemLabel, gridConstraints);

itemTextField.setPreferredSize(new Dimension(400, 25));

gridConstraints = new GridBagConstraints();

gridConstraints.gridx = 2;

gridConstraints.gridy = 0;

gridConstraints.gridwidth = 5;

gridConstraints.insets = new Insets(10, 0, 0, 10);

gridConstraints.anchor = GridBagConstraints.WEST;

getContentPane().add(itemTextField, gridConstraints);

itemTextField.addActionListener(new ActionListener ()

{

Public void actionPerformed(ActionEvent e)

{

itemTextFieldActionPerformed€;

}

});

locationLabel.setText(“Location”);

gridConstraints = new GridBagConstraints();

gridConstraints.gridx = 1;

gridConstraints.gridy = 1;

gridConstraints.insets = new Insets(10, 10, 0, 10);

gridConstraints.anchor = GridBagConstraints.EAST;

getContentPane().add(locationLabel, gridConstraints);

locationComboBox.setPreferredSize(new Dimension(270, 25));

locationComboBox.setFont(new Font(“Arial”, Font.PLAIN, 12));

locationComboBox.setEditable(true);

locationComboBox.setBackground(Color.WHITE);

gridConstraints = new GridBagConstraints();

gridConstraints.gridx = 2;

gridConstraints.gridy = 1;

gridConstraints.gridwidth = 3;

gridConstraints.insets = new Insets(10, 0, 0, 10);

gridConstraints.anchor = GridBagConstraints.WEST;

getContentPane().add(locationComboBox, gridConstraints);

locationComboBox.addActionListener(new ActionListener ()

{

Public void actionPerformed(ActionEvent e)

{

locationComboBoxActionPerformed€;

}

});

markedCheckBox.setText(“Marked?”);

markedCheckBox.setFocusable(false);

gridConstraints = new GridBagConstraints();

gridConstraints.gridx = 5;

gridConstraints.gridy = 1;

gridConstraints.insets = new Insets(10, 10, 0, 0);

gridConstraints.anchor = GridBagConstraints.WEST;

getContentPane().add(markedCheckBox, gridConstraints);

serialLabel.setText(“Serial Number”);

gridConstraints = new GridBagConstraints();

gridConstraints.gridx = 1;

gridConstraints.gridy = 2;

gridConstraints.insets = new Insets(10, 10, 0, 10);

gridConstraints.anchor = GridBagConstraints.EAST;

getContentPane().add(serialLabel, gridConstraints);

serialTextField.setPreferredSize(new Dimension(270, 25));

gridConstraints = new GridBagConstraints();

gridConstraints.gridx = 2;

gridConstraints.gridy = 2;

gridConstraints.gridwidth = 3;

gridConstraints.insets = new Insets(10, 0, 0, 10);

gridConstraints.anchor = GridBagConstraints.WEST;

getContentPane().add(serialTextField, gridConstraints);

serialTextField.addActionListener(new ActionListener ()

{

Public void actionPerformed(ActionEvent e)

{

serialTextFieldActionPerformed€;

}

});

priceLabel.setText(“Purchase Price”);

gridConstraints = new GridBagConstraints();

gridConstraints.gridx = 1;

gridConstraints.gridy = 3;

gridConstraints.insets = new Insets(10, 10, 0, 10);

gridConstraints.anchor = GridBagConstraints.EAST;

getContentPane().add(priceLabel, gridConstraints);

priceTextField.setPreferredSize(new Dimension(160, 25));

gridConstraints = new GridBagConstraints();

gridConstraints.gridx = 2;

gridConstraints.gridy = 3;

gridConstraints.gridwidth = 2;

gridConstraints.insets = new Insets(10, 0, 0, 10);

gridConstraints.anchor = GridBagConstraints.WEST;

getContentPane().add(priceTextField, gridConstraints);

priceTextField.addActionListener(new ActionListener ()

{

Public void actionPerformed(ActionEvent e)

{

priceTextFieldActionPerformed€;

}

});

dateLabel.setText(“Date Purchased”);

gridConstraints = new GridBagConstraints();

gridConstraints.gridx = 4;

gridConstraints.gridy = 3;

gridConstraints.insets = new Insets(10, 10, 0, 0);

gridConstraints.anchor = GridBagConstraints.WEST;

getContentPane().add(dateLabel, gridConstraints);

dateDateChooser.setPreferredSize(new Dimension(120, 25));

gridConstraints = new GridBagConstraints();

gridConstraints.gridx = 5;

gridConstraints.gridy = 3;

gridConstraints.gridwidth = 2;

gridConstraints.insets = new Insets(10, 0, 0, 10);

gridConstraints.anchor = GridBagConstraints.WEST;

getContentPane().add(dateDateChooser, gridConstraints);

dateDateChooser.addPropertyChangeListener(new PropertyChangeListener()

{

Public void propertyChange(PropertyChangeEvent e)

{

dateDateChooserPropertyChange€;

}

});

storeLabel.setText(“Store/Website”);

gridConstraints = new GridBagConstraints();

gridConstraints.gridx = 1;

gridConstraints.gridy = 4;

gridConstraints.insets = new Insets(10, 10, 0, 10);

gridConstraints.anchor = GridBagConstraints.EAST;

getContentPane().add(storeLabel, gridConstraints);

storeTextField.setPreferredSize(new Dimension(400, 25));

gridConstraints = new GridBagConstraints();

gridConstraints.gridx = 2;

gridConstraints.gridy = 4;

gridConstraints.gridwidth = 5;

gridConstraints.insets = new Insets(10, 0, 0, 10);

gridConstraints.anchor = GridBagConstraints.WEST;

getContentPane().add(storeTextField, gridConstraints);

storeTextField.addActionListener(new ActionListener ()

{

Public void actionPerformed(ActionEvent e)

{

storeTextFieldActionPerformed€;

}

});

noteLabel.setText(“Note”);

gridConstraints = new GridBagConstraints();

gridConstraints.gridx = 1;

gridConstraints.gridy = 5;

gridConstraints.insets = new Insets(10, 10, 0, 10);

gridConstraints.anchor = GridBagConstraints.EAST;

getContentPane().add(noteLabel, gridConstraints);

noteTextField.setPreferredSize(new Dimension(400, 25));

gridConstraints = new GridBagConstraints();

gridConstraints.gridx = 2;

gridConstraints.gridy = 5;

gridConstraints.gridwidth = 5;

gridConstraints.insets = new Insets(10, 0, 0, 10);

gridConstraints.anchor = GridBagConstraints.WEST;

getContentPane().add(noteTextField, gridConstraints);

noteTextField.addActionListener(new ActionListener ()

{

Public void actionPerformed(ActionEvent e)

{

noteTextFieldActionPerformed€;

}

});

photoLabel.setText(“Photo”);

gridConstraints = new GridBagConstraints();

gridConstraints.gridx = 1;

gridConstraints.gridy = 6;

gridConstraints.insets = new Insets(10, 10, 0, 10);

gridConstraints.anchor = GridBagConstraints.EAST;

getContentPane().add(photoLabel, gridConstraints);

photoTextArea.setPreferredSize(new Dimension(350, 35));

photoTextArea.setFont(new Font(“Arial”, Font.PLAIN, 12));

photoTextArea.setEditable(false);

photoTextArea.setLineWrap(true);

photoTextArea.setWrapStyleWord(true);

photoTextArea.setBackground(new Color(255, 255, 192));

photoTextArea.setBorder(BorderFactory.createLineBorder(Color.BLACK));

photoTextArea.setFocusable(false);

gridConstraints = new GridBagConstraints();

gridConstraints.gridx = 2;

gridConstraints.gridy = 6;

gridConstraints.gridwidth = 4;

gridConstraints.insets = new Insets(10, 0, 0, 10);

gridConstraints.anchor = GridBagConstraints.WEST;

getContentPane().add(photoTextArea, gridConstraints);

photoButton.setText(“…”);

gridConstraints = new GridBagConstraints();

gridConstraints.gridx = 6;

gridConstraints.gridy = 6;

gridConstraints.insets = new Insets(10, 0, 0, 10);

gridConstraints.anchor = GridBagConstraints.WEST;

getContentPane().add(photoButton, gridConstraints);

photoButton.addActionListener(new ActionListener ()

{

Public void actionPerformed(ActionEvent e)

{

photoButtonActionPerformed€;

}

});

searchPanel.setPreferredSize(new Dimension(240, 160));

searchPanel.setBorder(BorderFactory.createTitledBorder(“Item Search”));

searchPanel.setLayout(new GridBagLayout());

gridConstraints = new GridBagConstraints();

gridConstraints.gridx = 1;

gridConstraints.gridy = 7;

gridConstraints.gridwidth = 3;

gridConstraints.insets = new Insets(10, 0, 10, 0);

gridConstraints.anchor = GridBagConstraints.CENTER;

getContentPane().add(searchPanel, gridConstraints);

int x = 0, y = 0;

// create and position 26 buttons

For (int I = 0; I < 26; i++)

{

// create new button

searchButton[i] = new JButton();

// set text property

searchButton[i].setText(String.valueOf((char) (65 + i)));

searchButton[i].setFont(new Font(“Arial”, Font.BOLD, 12));

searchButton[i].setMargin(new Insets(-10, -10, -10, -10));

sizeButton(searchButton[i], new Dimension(37, 27));

searchButton[i].setBackground(Color.YELLOW);

searchButton[i].setFocusable(false);

gridConstraints = new GridBagConstraints();

gridConstraints.gridx = x;

gridConstraints.gridy = y;

searchPanel.add(searchButton[i], gridConstraints);

// add method

searchButton[i].addActionListener(new ActionListener ()

{

Public void actionPerformed(ActionEvent e)

{

searchButtonActionPerformed€;

}

});

X++;

// six buttons per row

If (x % 6 == 0)

{

X = 0;

Y++;

}

}

photoPanel.setPreferredSize(new Dimension(240, 160));

gridConstraints = new GridBagConstraints();

gridConstraints.gridx = 4;

gridConstraints.gridy = 7;

gridConstraints.gridwidth = 3;

gridConstraints.insets = new Insets(10, 0, 10, 10);

gridConstraints.anchor = GridBagConstraints.CENTER;

getContentPane().add(photoPanel, gridConstraints);

pack();

Dimension screenSize =

Toolkit.getDefaultToolkit().getScreenSize();

setBounds((int) (0.5 \* (screenSize.width – getWidth())), (int) (0.5 \* (screenSize.height –

getHeight())), getWidth(), getHeight());

int n;

// open file for entries

Try

{

BufferedReader inputFile = new BufferedReader(new FileReader(“inventory.txt”));

numberEntries =

Integer.valueOf(inputFile.readLine()).intValue();

If (numberEntries != 0)

{

For (int I = 0; I < numberEntries; i++)

{

myInventory[i] = new InventoryItem();

myInventory[i].description = inputFile.readLine();

myInventory[i].location = inputFile.readLine();

myInventory[i].serialNumber = inputFile.readLine();

myInventory[i].marked =

Boolean.valueOf(inputFile.readLine()).booleanValue();

myInventory[i].purchasePrice =

inputFile.readLine();

myInventory[i].purchaseDate = inputFile.readLine();

myInventory[i].purchaseLocation =

inputFile.readLine();

myInventory[i].note = inputFile.readLine();

myInventory[i].photoFile = inputFile.readLine();

}

}

// read in combo box elements

N = Integer.valueOf(inputFile.readLine()).intValue();

If (n != 0)

{

For (int I = 0; I < n; i++)

{

locationComboBox.addItem(inputFile.readLine());

}

}

inputFile.close();

currentEntry = 1;

showEntry(currentEntry);

}

Catch (Exception ex)

{

numberEntries = 0;

currentEntry = 0;

}

If (numberEntries == 0)

{

newButton.setEnabled(false);

deleteButton.setEnabled(false);

nextButton.setEnabled(false);

previousButton.setEnabled(false);

printButton.setEnabled(false);

}

}

Private void exitForm(WindowEvent evt)

{

If (JOptionPane.showConfirmDialog(null, “Any unsaved changes will be lost.\nAre you sure you want to exit?”, “Exit Program”, JOptionPane.YES\_NO\_OPTION, JOptionPane.QUESTION\_MESSAGE) == JOptionPane.NO\_OPTION)

Return;

// write entries back to file

Try

{

PrintWriter outputFile = new PrintWriter(new BufferedWriter(new

FileWriter(“inventory.txt”)));

outputFile.println(numberEntries);

if (numberEntries != 0)

{

For (int I = 0; I < numberEntries; i++)

{

outputFile.println(myInventory[i].description);

outputFile.println(myInventory[i].location);

outputFile.println(myInventory[i].serialNumber);

outputFile.println(myInventory[i].marked);

outputFile.println(myInventory[i].purchasePrice);

outputFile.println(myInventory[i].purchaseDate);

outputFile.println(myInventory[i].purchaseLocation);

outputFile.println(myInventory[i].note);

outputFile.println(myInventory[i].photoFile);

}

}

// write combo box entries

outputFile.println(locationComboBox.getItemCount());

if (locationComboBox.getItemCount() != 0)

{

For (int I = 0; I < locationComboBox.getItemCount(); i++)

outputFile.println(locationComboBox.getItemAt(i));

}

outputFile.close();

}

Catch (Exception ex)

{

}

System.exit(0);

}

Private void newButtonActionPerformed(ActionEvent e)

{

checkSave();

blankValues();

}

Private void deleteButtonActionPerformed(ActionEvent e)

{

If (JOptionPane.showConfirmDialog(null, “Are you sure you want to delete this item?”,

“Delete Inventory Item”, JOptionPane.YES\_NO\_OPTION,

JOptionPane.QUESTION\_MESSAGE) == JOptionPane.NO\_OPTION)

Return;

deleteEntry(currentEntry);

if (numberEntries == 0)

{

currentEntry = 0;

blankValues();

}

Else

{

currentEntry--;

if (currentEntry == 0)

currentEntry = 1;

showEntry(currentEntry);

}

}

Private void saveButtonActionPerformed(ActionEvent e)

{

// check for description

itemTextField.setText(itemTextField.getText().trim());

if (itemTextField.getText().equals(“”))

{

JOptionPane.showConfirmDialog(null, “Must have item description.”, “Error”,

JOptionPane.DEFAULT\_OPTION, JOptionPane.ERROR\_MESSAGE);

itemTextField.requestFocus();

return;

}

If (newButton.isEnabled())

{

// delete edit entry then resave

deleteEntry(currentEntry);

}

// capitalize first letter

String s = itemTextField.getText();

itemTextField.setText(s.substring(0, 1).toUpperCase() + s.substring(1));

numberEntries++;

// determine new current entry location based on description

currentEntry = 1;

if (numberEntries != 1)

{

Do

{

If

(itemTextField.getText().compareTo(myInventory[currentEntry – 1].description) < 0)

Break;

currentEntry++;

}

While (currentEntry < numberEntries);

}

// move all entries below new value down one position unless at end

If (currentEntry != numberEntries)

{

For (int I = numberEntries; I >= currentEntry + 1; i--)

{

myInventory[I – 1] = myInventory[I – 2];

myInventory[I – 2] = new InventoryItem();

}

}

myInventory[currentEntry – 1] = new InventoryItem();

myInventory[currentEntry – 1].description = itemTextField.getText();

myInventory[currentEntry – 1].location =

locationComboBox.getSelectedItem().toString();

myInventory[currentEntry – 1].marked = markedCheckBox.isSelected();

myInventory[currentEntry – 1].serialNumber = serialTextField.getText();

myInventory[currentEntry – 1].purchasePrice = priceTextField.getText();

myInventory[currentEntry – 1].purchaseDate =

dateToString(dateDateChooser.getDate());

myInventory[currentEntry – 1].purchaseLocation = storeTextField.getText();

myInventory[currentEntry – 1].photoFile = photoTextArea.getText();

myInventory[currentEntry – 1].note = noteTextField.getText();

showEntry(currentEntry);

if (numberEntries < maximumEntries)

newButton.setEnabled(true);

else

newButton.setEnabled(false);

deleteButton.setEnabled(true);

printButton.setEnabled(true);

}

Private void previousButtonActionPerformed(ActionEvent e)

{

checkSave();

currentEntry--;

showEntry(currentEntry);

}

Private void nextButtonActionPerformed(ActionEvent e)

{

checkSave();

currentEntry++;

showEntry(currentEntry);

}

Private void printButtonActionPerformed(ActionEvent e)

{

lastPage = (int) (1 + (numberEntries – 1) / entriesPerPage);

PrinterJob inventoryPrinterJob = PrinterJob.getPrinterJob();

inventoryPrinterJob.setPrintable(new InventoryDocument());

if (inventoryPrinterJob.printDialog())

{

Try

{

inventoryPrinterJob.print();

}

Catch (PrinterException ex)

{

JOptionPane.showConfirmDialog(null, ex.getMessage(), “Print Error”,

JOptionPane.DEFAULT\_OPTION, JOptionPane.ERROR\_MESSAGE);

}

}

}

Private void exitButtonActionPerformed(ActionEvent e)

{

exitForm(null);

}

Private void photoButtonActionPerformed(ActionEvent e)

{

JFileChooser openChooser = new JFileChooser();

openChooser.setDialogType(JFileChooser.OPEN\_DIALOG);

openChooser.setDialogTitle(“Open Photo File”);

openChooser.addChoosableFileFilter(new FileNameExtensionFilter(“Photo Files”,

“jpg”));

If (openChooser.showOpenDialog(this) == JFileChooser.APPROVE\_OPTION)

showPhoto(openChooser.getSelectedFile().toString());

}

Private void searchButtonActionPerformed(ActionEvent e)

{

Int I;

If (numberEntries == 0)

Return;

// search for item letter

String letterClicked = e.getActionCommand();

I = 0;

Do

{

If (myInventory[i].description.substring(0, 1).equals(letterClicked))

{

currentEntry = I + 1;

showEntry(currentEntry);

return;

}

I++;

}

While (I < numberEntries);

JOptionPane.showConfirmDialog(null, “No “ + letterClicked + “ inventory items.”,

“None Found”, JOptionPane.DEFAULT\_OPTION,

JOptionPane.INFORMATION\_MESSAGE);

}

Private void itemTextFieldActionPerformed(ActionEvent e)

{

locationComboBox.requestFocus();

}

Private void locationComboBoxActionPerformed(ActionEvent e)

{

// If in list – exit method

If (locationComboBox.getItemCount() != 0)

{

For (int I = 0; I < locationComboBox.getItemCount(); i++)

{

If (locationComboBox.getSelectedItem().toString().equals(locationComboBox.getItemAt(i).toString()))

{

serialTextField.requestFocus();

return;

}

}

}

// If not found, add to list box

locationComboBox.addItem(locationComboBox.getSelectedItem());

serialTextField.requestFocus();

}

Private void serialTextFieldActionPerformed(ActionEvent e)

{

priceTextField.requestFocus();

}

Private void priceTextFieldActionPerformed(ActionEvent e)

{

dateDateChooser.requestFocus();

}

Private void dateDateChooserPropertyChange(PropertyChangeEvent e)

{

storeTextField.requestFocus();

}

Private void storeTextFieldActionPerformed(ActionEvent e)

{

noteTextField.requestFocus();

}

Private void noteTextFieldActionPerformed(ActionEvent e)

{

photoButton.requestFocus();

}

Private void sizeButton(JButton b, Dimension d)

{

b.setPreferredSize(d);

b.setMinimumSize(d);

b.setMaximumSize(d);

}

Private void showEntry(int j)

{

// display entry j (1 to numberEntries)

itemTextField.setText(myInventory[j – 1].description);

locationComboBox.setSelectedItem(myInventory[j – 1].location);

markedCheckBox.setSelected(myInventory[j – 1].marked);

serialTextField.setText(myInventory[j – 1].serialNumber);

priceTextField.setText(myInventory[j – 1].purchasePrice);

dateDateChooser.setDate(stringToDate(myInventory[j – 1].purchaseDate));

storeTextField.setText(myInventory[j – 1].purchaseLocation);

noteTextField.setText(myInventory[j – 1].note);

showPhoto(myInventory[j – 1].photoFile);

nextButton.setEnabled(true);

previousButton.setEnabled(true);

if (j == 1)

previousButton.setEnabled(false);

if (j == numberEntries)

nextButton.setEnabled(false);

itemTextField.requestFocus();

}

Private Date stringToDate(String s)

{

Int m = Integer.valueOf(s.substring(0, 2)).intValue() – 1;

Int d = Integer.valueOf(s.substring(3, 5)).intValue();

Int y = Integer.valueOf(s.substring(6)).intValue() – 1900;

Return(new Date(y, m, d));

}

Private String dateToString(Date dd)

{

String yString = String.valueOf(dd.getYear() + 1900);

Int m = dd.getMonth() + 1;

String mString = new DecimalFormat(“00”).format(m);

Int d = dd.getDate();

String dString = new DecimalFormat(“00”).format(d);

Return(mString + “/” + dString + “/” + yString);

}

Private void showPhoto(String photoFile)

{

If (!photoFile.equals(“”))

{

Try

{

photoTextArea.setText(photoFile);

}

Catch (Exception ex)

{

photoTextArea.setText(“”);

}

}

Else

{

photoTextArea.setText(“”);

}

photoPanel.repaint();

}

Private void blankValues()

{

// blank input screen

newButton.setEnabled(false);

deleteButton.setEnabled(false);

saveButton.setEnabled(true);

previousButton.setEnabled(false);

nextButton.setEnabled(false);

printButton.setEnabled(false);

itemTextField.setText(“”);

locationComboBox.setSelectedItem(“”);

markedCheckBox.setSelected(false);

serialTextField.setText(“”);

priceTextField.setText(“”);

dateDateChooser.setDate(new Date());

storeTextField.setText(“”);

noteTextField.setText(“”);

photoTextArea.setText(“”);

photoPanel.repaint();

itemTextField.requestFocus();

}

Private void deleteEntry(int j)

{

// delete entry j

If (j != numberEntries)

{

// move all entries under j up one level

For (int I = j; I < numberEntries; i++)

{

myInventory[I – 1] = new InventoryItem();

myInventory[I – 1] = myInventory[i];

}

}

numberEntries--;

}

Private void checkSave()

{

Boolean edited = false;

If (!myInventory[currentEntry – 1].description.equals(itemTextField.getText()))

Edited = true;

Else if (!myInventory[currentEntry -1].location.equals(locationComboBox.getSelectedItem().toString()))edited = true;

Else if (myInventory[currentEntry – 1].marked != markedCheckBox.isSelected()) edited = true;

Else if (!myInventory[currentEntry – 1].serialNumber.equals(serialTextField.getText()))

Edited = true;

Else if (!myInventory[currentEntry – 1].purchasePrice.equals(priceTextField.getText()))

Edited = true;

Else if (!myInventory[currentEntry –

1].purchaseDate.equals(dateToString(dateDateChooser.getDate())))

Edited = true;

Else if (!myInventory[currentEntry –

1].purchaseLocation.equals(storeTextField.getText()))

Edited = true;

Else if (!myInventory[currentEntry – 1].note.equals(noteTextField.getText()))

Edited = true;

Else if (!myInventory[currentEntry – 1].photoFile.equals(photoTextArea.getText()))

Edited = true;

If (edited)

{

If (JOptionPane.showConfirmDialog(null, “You have edited this item. Do you want to save the changes?”, “Save Item”, JOptionPane.YES\_NO\_OPTION,JOptionPane.QUESTION\_MESSAGE) == JOptionPane.YES\_OPTION)

saveButton.doClick();

}

}

}

Class PhotoPanel extends JPanel

{

Public void paintComponent(Graphics g)

{

Graphics2D g2D = (Graphics2D) g;

Super.paintComponent(g2D);

// draw border

G2D.setPaint(Color.BLACK);

G2D.draw(new Rectangle2D.Double(0, 0, getWidth() – 1, getHeight() – 1));

// show photo

Image photoImage = new

ImageIcon(HomeInventory.photoTextArea.getText()).getImage();

Int w = getWidth();

Int h = getHeight();

Double rWidth = (double) getWidth() / (double) photoImage.getWidth(null);

Double rHeight = (double) getHeight() / (double) photoImage.getHeight(null);

If (rWidth > rHeight)

{

// leave height at display height, change width by amount height is changed

W = (int) (photoImage.getWidth(null) \* rHeight);

}

Else

{

// leave width at display width, change height by amount width is changed

H = (int) (photoImage.getHeight(null) \* rWidth);

}

// center in panel

G2D.drawImage(photoImage, (int) (0.5 \* (getWidth() – w)), (int) (0.5 \* (getHeight() –

h)), w, h, null);

g2D.dispose();

}

}

Class InventoryDocument implements Printable

{

Public int print(Graphics g, PageFormat pf, int pageIndex)

{

Graphics2D g2D = (Graphics2D) g;

If ((pageIndex + 1) > HomeInventory.lastPage)

{

Return NO\_SUCH\_PAGE;

}

Int I, iEnd;

// here you decide what goes on each page and draw it

// header

G2D.setFont(new Font(“Arial”, Font.BOLD, 14));

G2D.drawString(“Home Inventory Items – Page “ + String.valueOf(pageIndex + 1),

(int) pf.getImageableX(), (int) (pf.getImageableY() + 25));

// get starting y

Int dy = (int) g2D.getFont().getStringBounds(“S”,

G2D.getFontRenderContext()).getHeight();

Int y = (int) (pf.getImageableY() + 4 \* dy);

iEnd = HomeInventory.entriesPerPage \* (pageIndex + 1);

if (iEnd > HomeInventory.numberEntries)

iEnd = HomeInventory.numberEntries;

for (I = 0 + HomeInventory.entriesPerPage \* pageIndex; I < iEnd; i++)

{

// dividing line

Line2D.Double dividingLine = new

Line2D.Double(pf.getImageableX(), y, pf.getImageableX() + pf.getImageableWidth(), y);

G2D.draw(dividingLine);

Y += dy;

G2D.setFont(new Font(“Arial”, Font.BOLD, 12));

G2D.drawString(HomeInventory.myInventory[i].description, (int) pf.getImageableX(), y);

Y += dy;

G2D.setFont(new Font(“Arial”, Font.PLAIN, 12));

G2D.drawString(“Location: “ + HomeInventory.myInventory[i].location, (int)

(pf.getImageableX() + 25), y);

Y += dy;

If (HomeInventory.myInventory[i].marked)

G2D.drawString(“Item is marked with identifying information.”, (int)

(pf.getImageableX() + 25), y);

Else

G2D.drawString(“Item is NOT marked with identifying information.”, (int)

(pf.getImageableX() + 25), y);

Y += dy;

G2D.drawString(“Serial Number: “ +

HomeInventory.myInventory[i].serialNumber, (int) (pf.getImageableX() + 25), y);

Y += dy;

G2D.drawString(“Price: $” + HomeInventory.myInventory[i].purchasePrice + “,Purchased on: “ + HomeInventory.myInventory[i].purchaseDate, (int) (pf.getImageableX() +25), y);

Y += dy;

G2D.drawString(“Purchased at: “ +

HomeInventory.myInventory[i].purchaseLocation, (int) (pf.getImageableX() + 25), y);

Y += dy;

G2D.drawString(“Note: “ + HomeInventory.myInventory[i].note, (int)

(pf.getImageableX() + 25), y);

Y += dy;

Try

{

// maintain original width/height ratio

Image inventoryImage = new

ImageIcon(HomeInventory.myInventory[i].photoFile).getImage();

Double ratio = (double) (inventoryImage.getWidth(null)) / (double)

inventoryImage.getHeight(null);

g2D.drawImage(inventoryImage, (int) (pf.getImageableX() + 25), y, (int) (100 \*

ratio), 100, null);

}

Catch (Exception ex)

{

// have place to go in case image file doesn’t open

}

Y += 2 \* dy + 100;

}

Return PAGE\_EXISTS;

}

}