**Item Class**

public class Item implements Comparable<Object>

{

private String myName;

private String myCategory;

private int myStock;

private int myUsage;

private String myComments;

public Item(String name, String type, int stock,

int usage, String comments)

{

myName = name;

myCategory = type;

myStock = stock;

myUsage = usage;

myComments = comments;

}

public String getName()

{

return myName;

}

public void setName(String name)

{

this.myName = name;

}

public String getCategory()

{

return myCategory;

}

public void setCategory(String category)

{

this.myCategory= category;

}

public int getStock()

{

return myStock;

}

public void setStock(int stock)

{

this.myStock = stock;

}

public int getUsage()

{

return myUsage;

}

public void setUsage(int usage)

{

this.myUsage = usage;

}

public String getComments()

{

return myComments;

}

public void setComments(String comments)

{

this.myComments = comments;

}

public boolean isLow()

{

if(myStock <= myUsage)

return true;

else

return false;

}

public int compareTo(Object otherObject)

{

Item other = (Item)otherObject;

return myName.compareToIgnoreCase(other.getName());

}

public boolean equals(Object otherObject)

{

return this.compareTo(otherObject) == 0;

}

public boolean containsSpecific(String text)

{

if(myName.toLowerCase().equals(text.toLowerCase()))

return true;

else if(myCategory.toLowerCase().equals(text.toLowerCase()))

return true;

else if(Integer.toString(myStock).toLowerCase().equals(text.toLowerCase()))

return true;

else if(Integer.toString(myUsage).toLowerCase().equals(text.toLowerCase()))

return true;

else if(myComments.toLowerCase().equals(text.toLowerCase()))

return true;

else

return false;

}

public boolean containsGeneric(String text)

{

if(myName.toLowerCase().contains(text.toLowerCase()))

return true;

else if(myCategory.toLowerCase().contains(text.toLowerCase()))

return true;

else if(Integer.toString(myStock).toLowerCase().contains(text.toLowerCase()))

return true;

else if(Integer.toString(myUsage).toLowerCase().contains(text.toLowerCase()))

return true;

else if(myComments.toLowerCase().contains(text.toLowerCase()))

return true;

else

return false;

}

public String toString()

{

return "Name=" + myName + ", Category=" + myCategory + ", Stock=" + myStock +

", Approximate Used Weekly=" + myUsage + ", isLow=" + isLow() + ", Comments=" + myComments ;

}

}

**Methods Class**

import java.io.\*;

import java.util.\*;

import java.util.Scanner;

//import javax.swing.JFrame;

//import javax.swing.JOptionPane;

import java.io.File;

public class Methods

{

private ArrayList<Item> myList;

public Methods(ArrayList<Item> itemList)

{

myList = itemList;

}

public ArrayList<Item> loadData()

{

Scanner inFile;

String fileName = "ItemList.txt";

String name;

String type;

int stock;

int usage;

String comments;

try

{

File file = new File(fileName);

file.createNewFile();

}

catch(IOException e)

{

System.out.println("Error could not create file " + e.getMessage());

}

try

{

inFile = new Scanner(new File(fileName));

while(inFile.hasNextLine())

{

name = inFile.nextLine();

type = inFile.nextLine();

stock = Integer.parseInt(inFile.nextLine());

usage = Integer.parseInt(inFile.nextLine());

comments = inFile.nextLine();

inFile.nextLine();

myList.add(new Item(name, type, stock, usage, comments));

}

inFile.close();

}

catch(IOException i)

{

System.out.println("Error could not find file " + i.getMessage());

}

return myList;

}

public void mergeSort(ArrayList <Item> list, int first, int last)

{

if(first == last)

{

return;

}

else if (last-first == 1)

{

if(list.get(first).compareTo(list.get(last)) > 0)

{

ArrayList<Item> var = new ArrayList<Item>();

var.add(list.get(first));

list.set(first, list.get(last));

list.set(last, var.get(0));

}

}

else

{

int mid = (first + last) / 2;

mergeSort(list, first, mid);

mergeSort(list, mid+1, last);

merge(list, first, mid, last);

}

}

// creates a copy of ArrayList list called temp, and uses temp’s values to properly merge(sort) list

// from first to last

public void merge(ArrayList<Item> list, int first, int mid, int last)

{

ArrayList<Item> merged = new ArrayList<Item>();

int index1 = first, index2 = mid + 1;

for(int i = first; i <= last; i++)

{

if (index1 > mid)

{

merged.add(list.get(index2));

index2++;

}

else if (index2 > last)

{

merged.add(list.get(index1));

index1++;

}

else if (list.get(index1).compareTo(list.get(index2)) < 0)

{

merged.add(list.get(index1));

index1++;

}

else

{

merged.add(list.get(index2));

index2++;

}

}

int mergedIndex = 0;

for (int j = first; j <= last; j++)

{

list.set(j,merged.get(mergedIndex));

mergedIndex++;

}

}

public void reversemergeSort(ArrayList <Item> list, int first, int last)

{

if(first == last)

{

return;

}

else if (last-first == 1)

{

if(list.get(first).compareTo(list.get(last)) < 0)

{

ArrayList<Item> var = new ArrayList<Item>();

var.add(list.get(first));

list.set(first, list.get(last));

list.set(last, var.get(0));

}

}

else

{

int mid = (first + last) / 2;

reversemergeSort(list, first, mid);

reversemergeSort(list, mid+1, last);

reversemerge(list, first, mid, last);

}

}

// creates a copy of ArrayList list called temp, and uses temp’s values to properly merge(sort) list

// from first to last

public void reversemerge(ArrayList<Item> list, int first, int mid, int last)

{

ArrayList<Item> merged = new ArrayList<Item>();

int index1 = first, index2 = mid + 1;

for(int i = first; i <= last; i++)

{

if (index1 > mid)

{

merged.add(list.get(index2));

index2++;

}

else if (index2 > last)

{

merged.add(list.get(index1));

index1++;

}

else if (list.get(index1).compareTo(list.get(index2)) > 0)

{

merged.add(list.get(index1));

index1++;

}

else

{

merged.add(list.get(index2));

index2++;

}

}

int mergedIndex = 0;

for (int j = first; j <= last; j++)

{

list.set(j,merged.get(mergedIndex));

mergedIndex++;

}

}

public ArrayList<Item> organizeList()

{

for (int outer = 1; outer < myList.size(); outer++)

{

int position = outer;

Item key = myList.get(position);

while (position > 0 && myList.get(position - 1).compareTo(key) >= 0)

{

myList.set(position, myList.get(position - 1));

position--;

}

myList.set(position, key);

}

return myList;

}

public void organizeCategory(ArrayList<Item> array)

{

for (int outer = 1; outer < array.size(); outer++)

{

int position = outer;

Item key = array.get(position);

while (position > 0 && array.get(position - 1).getCategory().compareTo(key.getCategory()) > 0)

{

array.set(position, array.get(position - 1));

position--;

}

array.set(position, key);

}

//return myList;

}

public void organizeStock(ArrayList<Item> array)

{

for (int outer = 1; outer < array.size(); outer++)

{

int position = outer;

Item key = array.get(position);

while (position > 0 && array.get(position - 1).getStock() >= (key.getStock()))

{

array.set(position, array.get(position - 1));

position--;

}

array.set(position, key);

}

//return myList;

}

public void organizeUsage(ArrayList<Item> array)

{

for (int outer = 1; outer < array.size(); outer++)

{

int position = outer;

Item key = myList.get(position);

while (position > 0 && array.get(position - 1).getUsage() >= (key.getUsage()))

{

array.set(position, array.get(position - 1));

position--;

}

array.set(position, key);

}

//return myList;

}

public void modifyFile()

{

try

{

PrintWriter fw = new PrintWriter( new FileWriter("ItemList.txt"));

for( int a = 0; a < myList.size(); a++)

{

fw.println(myList.get(a).getName());

fw.println(myList.get(a).getCategory());

fw.println(myList.get(a).getStock());

fw.println(myList.get(a).getUsage());

fw.println(myList.get(a).getComments());

fw.println();

}

fw.close();

}

catch(Exception e)

{

//JFrame frame = new JFrame();

//JOptionPane.showMessageDialog(frame, "File not found");

System.out.println(e);

}

}

public Item returnAll(ArrayList<Item> insert)

{

Item value = null;

for(int i = 0; i < myList.size(); i++)

{

value = insert.get(i);

}

return value;

}

public ArrayList<Item> addItem(String name, String type, int stock,

int usage, String comments)

{

Item added = new Item(name, type, stock, usage, comments);

myList.add(added);

//mergeSort(myList, 0, myList.size());

organizeList();

return myList;

}

public ArrayList<Item> editItem(String old, String name, String type, int stock,

int usage, String comments )

{

for(int index = 0; index < myList.size(); index++)

{

if(myList.get(index).getName().equals(old))

myList.set(index, new Item( name, type, stock, usage, comments));

}

//mergeSort(myList, 0, myList.size());

organizeList();

return myList;

}

public ArrayList<Item> searchItems(String input) //int stock, String type,

//int usage, String comments)

{

ArrayList<Item> selected = new ArrayList<Item>();

for(int i = 0; i < myList.size(); i++)

{

if( myList.get(i).getName().indexOf(input) > -1 ||

myList.get(i).getStock() == Integer.parseInt(input) ||

myList.get(i).getCategory().indexOf(input) > -1 ||

myList.get(i).getUsage() == Integer.parseInt(input) ||

myList.get(i).getComments().indexOf(input) > -1)

selected.add(myList.get(i));

}

return selected;

}

public ArrayList<Item> removeItem(String name)

{

for(int i = 0; i < myList.size(); i++)

{

if( myList.get(i).getName().equals(name))

{

myList.remove(i);

i--;

}

}

return myList;

}

}

**MainMenu Class**

import java.io.\*;

import java.util.\*;

import java.awt.\*;

import javax.swing.\*;

import java.awt.BorderLayout;

import java.awt.EventQueue;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.border.EmptyBorder;

import javax.swing.event.ListSelectionEvent;

import javax.swing.event.ListSelectionListener;

import javax.swing.table.DefaultTableModel;

import javax.swing.table.TableColumn;

import net.miginfocom.swing.MigLayout;

import javax.swing.JButton;

import javax.swing.BoxLayout;

import javax.swing.DefaultListModel;

import javax.swing.ImageIcon;

import javax.swing.JScrollPane;

import javax.swing.JRadioButton;

import javax.swing.JTextField;

import javax.swing.ListModel;

import javax.swing.ListSelectionModel;

import java.awt.Window.Type;

import javax.swing.SwingConstants;

import java.awt.event.ActionListener;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.awt.event.ActionEvent;

import javax.swing.JLabel;

import javax.swing.JList;

import javax.swing.JScrollBar;

import javax.swing.JRadioButtonMenuItem;

import javax.swing.JMenuBar;

import javax.swing.JMenuItem;

import javax.swing.JOptionPane;

import javax.swing.AbstractListModel;

import javax.swing.JTable;

import javax.swing.JComboBox;

import java.awt.Font;

import java.awt.Color;

import java.awt.Desktop;

import javax.swing.ScrollPaneConstants;

import javax.swing.border.LineBorder;

// USE MERGESORT

public class MainMenu extends JFrame

{

private JPanel contentPane;

private JTextField textSearch;

private Methods functions;

private ArrayList<Item> items;

private JTable table;

/\*\*

\* Launch the application.

\*/

public static void main(String[] args)

{

EventQueue.invokeLater(new Runnable()

{

public void run() {

try {

MainMenu frame = new MainMenu();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the frame.

\*/

public MainMenu()

{

setForeground(Color.WHITE);

items = new ArrayList<Item>();

functions = new Methods(items);

items = functions.loadData();

setResizable(false);

setTitle("ITEM DATABASE");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setBounds(0, -17, 950, 652);

contentPane = new JPanel();

contentPane.setBackground(new Color(245, 222, 179));

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

contentPane.setLayout(null);

JScrollPane scrollPane = new JScrollPane();

scrollPane.setBounds(20, 10, 900, 405);

contentPane.add(scrollPane);

DefaultTableModel model = new DefaultTableModel();

model.addColumn("Name");

model.addColumn("Category");

model.addColumn("Stock");

model.addColumn("Weekly Usage");

model.addColumn("Comments");

for(int index = 0; index < items.size(); index++)

{

Item value = items.get(index);

model.addRow(new Object[] {value.getName(), value.getCategory(), value.getStock(), value.getUsage(), value.getComments() });

}

table = new JTable(model);

table.setSelectionMode(ListSelectionModel.SINGLE\_INTERVAL\_SELECTION);

table.getTableHeader().setReorderingAllowed(false);

table.setDragEnabled(true);

//table.setAutoResizeMode(JTable.AUTO\_RESIZE\_OFF);

table.getColumnModel().getColumn(0).setPreferredWidth(200);

table.getColumnModel().getColumn(1).setPreferredWidth(100);

table.getColumnModel().getColumn(2).setPreferredWidth(60);

table.getColumnModel().getColumn(3).setPreferredWidth(100);

table.getColumnModel().getColumn(4).setPreferredWidth(400);

scrollPane.setViewportView(table);

textSearch = new JTextField();

textSearch.setBounds(20, 502, 317, 27);

contentPane.add(textSearch);

textSearch.setColumns(10);

JButton btnAdd = new JButton("ADD");

btnAdd.setToolTipText("Add an Item to the Main Menu\r\n");

btnAdd.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e)

{

AddWindow addFunction = new AddWindow(functions, items, MainMenu.this, model);

//sortButton.setSelected(false);

MainMenu.this.setVisible(false);

addFunction.setVisible(true);

}

});

btnAdd.setBounds(20, 426, 140, 50);

contentPane.add(btnAdd);

JButton btnEdit = new JButton("EDIT");

btnEdit.setToolTipText("Select an Item from the table to Edit");

btnEdit.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e)

{

if(items.size() == 0)

{

JOptionPane.showMessageDialog(null,"LIST IS EMPTY, NO ITEMS TO EDIT");

}

else if(table.getSelectedRow() == -1)

{

JOptionPane.showMessageDialog(null,"PLEASE SELECT A ROW TO EDIT");

}

else

{

int row = table.getSelectedRow();

int itemIndex = 0;

String name = (String)model.getValueAt(row, 0);

String category = (String)model.getValueAt(row, 1);

int stock = (int)model.getValueAt(row, 2);

int usage = (int)model.getValueAt(row, 3);

String comments = (String)model.getValueAt(row, 4);

Item rowValue = new Item(name,category,stock,usage,comments);

for(int index = 0; index < items.size(); index++)

{

if( items.get(index).compareTo(rowValue) == 0)

{

itemIndex = index;

}

}

EditWindow editFunction = new EditWindow(row, itemIndex, functions, items, MainMenu.this, model);

//sortButton.setSelected(false);

MainMenu.this.setVisible(false);

editFunction.setVisible(true);

}

}

});

btnEdit.setBounds(197, 426, 140, 50);

contentPane.add(btnEdit);

JButton btnDelete = new JButton("DELETE");

btnDelete.setToolTipText("Select an Item from the table to Delete");

btnDelete.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent arg0)

{

if(items.size() == 0)

{

JOptionPane.showMessageDialog(null,"LIST IS EMPTY, NO ITEMS TO DELETE");

}

else if(table.getSelectedRow() == -1)

{

JOptionPane.showMessageDialog(null,"PLEASE SELECT A ROW TO DELETE");

}

else

{

int row = table.getSelectedRow();

String name = (String)model.getValueAt(row, 0);

String category = (String)model.getValueAt(row, 1);

int stock = (int)model.getValueAt(row, 2);

int usage = (int)model.getValueAt(row, 3);

String comments = (String)model.getValueAt(row, 4);

int prompt = JOptionPane.showConfirmDialog(null, "Are Sure You Want to Delete Item? \n\nName: " + name + "\nCategory: " + category + "\nStock: " + stock + "\nUsage: " + usage + "\nComments: " + comments

, "DOUBLE CHECK DELETE PROMPT", JOptionPane.YES\_NO\_OPTION);

if(prompt == JOptionPane.YES\_OPTION)

{

Item rowValue = new Item(name,category,stock,usage,comments);

for(int index = 0; index < items.size(); index++)

{

if( items.get(index).compareTo(rowValue) == 0)

{

functions.removeItem(items.get(index).getName());

functions.modifyFile();

functions.organizeList();

model.removeRow(row);

}

}

}

//DeleteWindow.this.setVisible(false);

//main.setVisible(true);

//DeleteWindow deleteFunction = new DeleteWindow(row, functions, items, MainMenu.this, model);

//sortButton.setSelected(false);

//MainMenu.this.setVisible(false);

//deleteFunction.setVisible(true);

}

}

});

btnDelete.setBounds(374, 426, 140, 50);

contentPane.add(btnDelete);

JRadioButton rdbtnSearch = new JRadioButton("SPECIFIC SEARCH");

rdbtnSearch.setToolTipText("Searches using full Words and not just fragments. ");

rdbtnSearch.setBackground(new Color(245, 222, 179));

rdbtnSearch.setBounds(374, 564, 140, 20);

contentPane.add(rdbtnSearch);

JButton btnSearch = new JButton("SEARCH ");

btnSearch.setToolTipText("Searches for an Item based off text entered in Textbox to the Left");

btnSearch.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e)

{

String toSearch = textSearch.getText();

if(items.size() == 0)

{

JOptionPane.showMessageDialog(null,"LIST IS EMPTY, NO ITEMS TO SEARCH");

}

else if(toSearch.length() <= 0)

{

JOptionPane.showMessageDialog(null, "PLEASE ENTER TEXT TO SEARCH");

}

else

{

model.setRowCount(0);

for(int index = 0; index < items.size(); index++)

{

Item value = items.get(index);

if(rdbtnSearch.isSelected())

{

if(value.containsSpecific(toSearch))

{

model.addRow(new Object[] {value.getName(), value.getCategory(), value.getStock(), value.getUsage(), value.getComments() });

}

}

else

{

if(value.containsGeneric(toSearch))

{

model.addRow(new Object[] {value.getName(), value.getCategory(), value.getStock(), value.getUsage(), value.getComments() });

}

}

}

}

}

});

btnSearch.setBounds(374, 490, 140, 50);

contentPane.add(btnSearch);

JComboBox<String> sortList = new JComboBox<String>();

sortList.addItem("BY NAME ALPHABETICAL (A-Z)");

sortList.addItem("BY NAME REVERSE ALPHABETICAL (Z-A)");

sortList.addItem("LOW STOCKED ITEMS");

sortList.addItem("BY CATEGORY");

sortList.addItem("BY STOCK (INCREASING)");

sortList.addItem("BY USAGE (INCREASING)");

sortList.setSelectedIndex(0);

sortList.addActionListener (new ActionListener () {

public void actionPerformed(ActionEvent e)

{

if(items.size() == 0)

{

JOptionPane.showMessageDialog(null,"LIST IS EMPTY, NO ITEMS TO SORT");

}

else

{

int sorted = sortList.getSelectedIndex();

if(sorted == 0)

{

model.setRowCount(0);

for(int index = 0; index < items.size(); index++)

{

Item value = items.get(index);

model.addRow(new Object[] {value.getName(), value.getCategory(), value.getStock(), value.getUsage(), value.getComments() });

}

}

else if(sorted == 1)

{

model.setRowCount(0);

ArrayList<Item> reverseAlphabetical = new ArrayList<Item>();

for(int index = 0; index < items.size(); index++)

{

Item values = items.get(index);

reverseAlphabetical.add(values);

}

functions.reversemergeSort(reverseAlphabetical, 0, reverseAlphabetical.size()-1);

for(int index = 0; index < reverseAlphabetical.size(); index++)

{

Item wholeList = reverseAlphabetical.get(index);

model.addRow(new Object[] {wholeList.getName(), wholeList.getCategory(), wholeList.getStock(), wholeList.getUsage(), wholeList.getComments() });

}

}

else if(sorted == 2)

{

ArrayList<Item> lowItems = new ArrayList<Item>();

for(Item value : items)

{

if(value.isLow())

lowItems.add(value);

}

model.setRowCount(0);

for( int index = 0; index < lowItems.size(); index++)

{

Item low = lowItems.get(index);

model.addRow(new Object[] {low.getName(), low.getCategory(), low.getStock(), low.getUsage(), low.getComments() });

}

}

else if(sorted == 3)

{

model.setRowCount(0);

ArrayList<Item> category = new ArrayList<Item>();

for(int index = 0; index < items.size(); index++)

{

Item values = items.get(index);

category.add(values);

}

functions.organizeCategory(category);

for(int index = 0; index < category.size(); index++)

{

Item wholeList = category.get(index);

model.addRow(new Object[] {wholeList.getName(), wholeList.getCategory(), wholeList.getStock(), wholeList.getUsage(), wholeList.getComments() });

}

}

else if(sorted == 4)

{

model.setRowCount(0);

ArrayList<Item> stock = new ArrayList<Item>();

for(int index = 0; index < items.size(); index++)

{

Item values = items.get(index);

stock.add(values);

}

functions.organizeStock(stock);

for(int index = 0; index < stock.size(); index++)

{

Item wholeList = stock.get(index);

model.addRow(new Object[] {wholeList.getName(), wholeList.getCategory(), wholeList.getStock(), wholeList.getUsage(), wholeList.getComments() });

}

}

else //if(sorted == 5)

{

model.setRowCount(0);

ArrayList<Item> usage = new ArrayList<Item>();

for(int index = 0; index < items.size(); index++)

{

Item values = items.get(index);

usage.add(values);

}

functions.organizeUsage(usage);

for(int index = 0; index < usage.size(); index++)

{

Item wholeList = usage.get(index);

model.addRow(new Object[] {wholeList.getName(), wholeList.getCategory(), wholeList.getStock(), wholeList.getUsage(), wholeList.getComments() });

}

}

}

}

});

sortList.setBounds(603, 426, 317, 20);

contentPane.add(sortList);

JButton btnRefresh = new JButton("REFRESH MAIN MENU");

btnRefresh.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0)

{

model.setRowCount(0);

for(int index = 0; index < items.size(); index++)

{

Item value = items.get(index);

model.addRow(new Object[] {value.getName(), value.getCategory(), value.getStock(), value.getUsage(), value.getComments() });

}

sortList.setSelectedIndex(0);

textSearch.setText("");

rdbtnSearch.setSelected(false);

}

});

btnRefresh.setBounds(20, 563, 317, 50);

contentPane.add(btnRefresh);

JButton btnTxt = new JButton("CREATE NEW PRINTABLE TEXT FILE");

btnTxt.setToolTipText("Opens viewable textfile sorted according to the combolist above, which the user can then print.");

btnTxt.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0)

{

if(model.getRowCount() == 0)

{

JOptionPane.showMessageDialog(null,"MAIN MENU IS EMPTY, NO ITEMS TO RETRACT");

}

else

{

String input = JOptionPane.showInputDialog("Please Label the Textfile");

if(input.length() == 0)

{

JOptionPane.showMessageDialog(null,"Please Label the Textfile");

}

else //if(input != null)

{

String textFileName = input + ".txt";

String name = "";

String category = "";

int stock = 0;

int usage = 0;

String comments = "";

ArrayList<Item> currentMenu = new ArrayList<Item>();

for( int index = 0; index < model.getRowCount(); index++)

{

name = (String)model.getValueAt(index, 0);

category = (String)model.getValueAt(index, 1);

stock = (int)model.getValueAt(index, 2);

usage = (int)model.getValueAt(index, 3);

comments = (String)model.getValueAt(index, 4);

Item temp = new Item(name, category, stock, usage, comments);

currentMenu.add(temp);

}

try

{

File newFile = new File(textFileName);

FileWriter write = new FileWriter(newFile);

PrintWriter editor = new PrintWriter(write);

for( int index = 0; index < currentMenu.size(); index++)

{

editor.println("NAME: " + currentMenu.get(index).getName());

editor.println("CATEGORY: " + currentMenu.get(index).getCategory());

editor.println("STOCK: " + currentMenu.get(index).getStock());

editor.println("USAGE: " + currentMenu.get(index).getUsage());

editor.println("COMMENTS: " + currentMenu.get(index).getComments());

editor.println();

}

editor.close();

}

catch(Exception e)

{

JFrame frame = new JFrame();

JOptionPane.showMessageDialog(frame, "File not found");

//System.out.println(e);

}

try

{

Desktop.getDesktop().open(new java.io.File(textFileName));

}

catch(Exception e)

{

e.printStackTrace();

}

}

}

}

});

btnTxt.setBounds(603, 563, 317, 50);

contentPane.add(btnTxt);

//JLabel imageLogo = new JLabel("");

//imageLogo.setIcon(new ImageIcon("savouryfusiongrilllogo3.png"));

//imageLogo.setBounds(633, 457, 250, 95);

//contentPane.add(imageLogo);

}

}

**AddWindow Class**

import java.awt.BorderLayout;

import java.awt.EventQueue;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.border.EmptyBorder;

import javax.swing.table.DefaultTableModel;

import javax.swing.GroupLayout;

import javax.swing.GroupLayout.Alignment;

import javax.swing.JButton;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import java.awt.Font;

import javax.swing.SwingConstants;

import javax.swing.JTextField;

import java.awt.event.ActionListener;

import java.util.ArrayList;

import java.awt.event.ActionEvent;

import java.awt.Color;

import java.awt.Component;

import java.awt.Window.Type;

import java.awt.Dialog.ModalExclusionType;

import javax.swing.JComboBox;

public class AddWindow extends JFrame

{

/\*\*

\*

\*/

//private Methods functions;

//private MainMenu list;

//private Item added;

private JPanel contentPane;

private JTextField txtName;

private JComboBox<String> txtCategory;

private JTextField txtStock;

private JTextField txtUsage;

private JTextField txtComments;

/\*\*

\* Launch the application.

\*/

public static void main(String[] args)

{

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

AddWindow frame = new AddWindow(null, null, null, null);

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the frame.

\*/

public AddWindow(Methods list, ArrayList<Item> menu, MainMenu main, DefaultTableModel model)

{

setFont(new Font("Dialog", Font.PLAIN, 33));

setResizable(false);

setTitle("ADD ITEMS");

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

setBounds(100, 100, 454, 399);

contentPane = new JPanel();

contentPane.setBackground(new Color(255, 255, 204));

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

contentPane.setLayout(null);

JLabel lblName = new JLabel("Name:");

lblName.setBounds(35, 25, 76, 14);

contentPane.add(lblName);

JLabel lblUsage = new JLabel("Weekly Usage:");

lblUsage.setBounds(35, 154, 112, 14);

contentPane.add(lblUsage);

JLabel lblComments = new JLabel("Comments:");

lblComments.setBounds(35, 239, 76, 14);

contentPane.add(lblComments);

JLabel lblStock = new JLabel("Stock:");

lblStock.setBounds(35, 110, 76, 14);

contentPane.add(lblStock);

JLabel lblCategory = new JLabel("Category:");

lblCategory.setBounds(35, 66, 92, 14);

contentPane.add(lblCategory);

txtName = new JTextField();

txtName.setBounds(157, 22, 275, 20);

contentPane.add(txtName);

txtName.setColumns(10);

txtCategory = new JComboBox<String>();

txtCategory.setToolTipText("");

txtCategory.addItem("Meat");

txtCategory.addItem("Vegetable");

txtCategory.addItem("Garnishes");

txtCategory.addItem("Seasoning");

txtCategory.addItem("Sauce");

txtCategory.addItem("Miscellaneous");

txtCategory.setSelectedIndex(-1);

txtCategory.setBounds(157, 60, 275, 20);

contentPane.add(txtCategory);

txtStock = new JTextField();

txtStock.setColumns(10);

txtStock.setBounds(157, 107, 275, 20);

contentPane.add(txtStock);

txtUsage = new JTextField();

txtUsage.setColumns(10);

txtUsage.setBounds(157, 151, 275, 20);

contentPane.add(txtUsage);

txtComments = new JTextField();

txtComments.setHorizontalAlignment(SwingConstants.LEFT);

txtComments.setColumns(10);

txtComments.setBounds(157, 218, 275, 57);

contentPane.add(txtComments);

JLabel nameError = new JLabel("");

nameError.setForeground(Color.RED);

nameError.setBounds(157, 42, 275, 14);

contentPane.add(nameError);

JLabel categoryError = new JLabel("");

categoryError.setForeground(Color.RED);

categoryError.setBounds(157, 82, 275, 14);

contentPane.add(categoryError);

JLabel stockError = new JLabel("");

stockError.setForeground(Color.RED);

stockError.setBounds(157, 126, 275, 14);

contentPane.add(stockError);

JLabel usageError = new JLabel("");

usageError.setForeground(Color.RED);

usageError.setBounds(157, 172, 275, 14);

contentPane.add(usageError);

//JLabel commentsError = new JLabel("");

//commentsError.setForeground(Color.RED);

//commentsError.setBounds(157, 277, 275, 14);

//contentPane.add(commentsError);

JButton btnAdd = new JButton("ADD");

btnAdd.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0)

{

if( txtName.getText().length() == 0)

{

nameError.setText("Empty Slot: Please Enter Name");

}

else

{

nameError.setText("");

if( txtCategory.getSelectedIndex() == -1)

{

categoryError.setText("Empty Slot: Please Select a Category");

}

else

{

categoryError.setText("");

try

{

Integer.parseInt(txtStock.getText());

if( Integer.parseInt(txtStock.getText()) < 0 )//|| Integer.parseInt(txtStock.getText()) > 200)

{

stockError.setText("Please Enter in a Positive Integer");

}

else

{

stockError.setText("");

try

{

Integer.parseInt(txtUsage.getText());

if( Integer.parseInt(txtUsage.getText()) < 0 )//|| Integer.parseInt(txtUsage.getText()) > 200)

{

usageError.setText("Please Enter in a Positive Integer");

}

else

{

usageError.setText("");

if( txtComments.getText().length() == 0)

{

int prompt = JOptionPane.showConfirmDialog(null, "Comments Section Blank, Leave Blank?", "COMMENT CHECKER", JOptionPane.YES\_NO\_OPTION);

if(prompt == JOptionPane.YES\_OPTION)

{

try

{

list.addItem(txtName.getText(), txtCategory.getSelectedItem().toString(), Integer.parseInt(txtStock.getText()),

Integer.parseInt(txtUsage.getText()), txtComments.getText());

list.modifyFile();

list.organizeList();

model.addRow(new Object[] {txtName.getText(), txtCategory.getSelectedItem().toString(), Integer.parseInt(txtStock.getText()),

Integer.parseInt(txtUsage.getText()), txtComments.getText() });

AddWindow.this.setVisible(false);

main.setVisible(true);

}

catch(Exception e)

{

JOptionPane.showMessageDialog(null,"Error!!!");

}

}

}

else

{

try

{

list.addItem(txtName.getText(), txtCategory.getSelectedItem().toString(), Integer.parseInt(txtStock.getText()),

Integer.parseInt(txtUsage.getText()), txtComments.getText());

list.modifyFile();

list.organizeList();

model.addRow(new Object[] {txtName.getText(), txtCategory.getSelectedItem().toString(), Integer.parseInt(txtStock.getText()),

Integer.parseInt(txtUsage.getText()), txtComments.getText() });

AddWindow.this.setVisible(false);

main.setVisible(true);

}

catch(Exception e)

{

JOptionPane.showMessageDialog(null,"Error!!!");

}

}

}

}

catch (NumberFormatException e)

{

usageError.setText("Please Enter in a Valid Integer");

}

}

}

catch (NumberFormatException e)

{

stockError.setText("Please Enter in a Valid Integer");

}

}

}

}

});

btnAdd.setBounds(50, 300, 150, 50);

contentPane.add(btnAdd);

JButton btnCancel = new JButton("CANCEL");

btnCancel.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0)

{

AddWindow.this.setVisible(false);

main.setVisible(true);

}

});

btnCancel.setBounds(250, 300, 150, 50);

contentPane.add(btnCancel);

}

}

**EditWindow Class**

import java.awt.\*;

import java.util.\*;

import javax.swing.\*;

import java.awt.BorderLayout;

import java.awt.EventQueue;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.border.EmptyBorder;

import javax.swing.table.DefaultTableModel;

import javax.swing.GroupLayout;

import javax.swing.GroupLayout.Alignment;

import javax.swing.JButton;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import java.awt.Font;

import javax.swing.SwingConstants;

import javax.swing.JTextField;

import java.awt.event.ActionListener;

import java.util.ArrayList;

import java.awt.event.ActionEvent;

import java.awt.Color;

import java.awt.Component;

import java.awt.Window.Type;

import java.awt.Dialog.ModalExclusionType;

import javax.swing.JComboBox;

public class EditWindow extends JFrame

{

/\*\*

\*

\*/

//private Methods functions;

//private MainMenu list;

//private Item added;

private JPanel contentPane;

private JTextField txtName;

private JComboBox<String> txtItems;

private JComboBox<String> txtCategory;

private JTextField txtStock;

private JTextField txtUsage;

private JTextField txtComments;

/\*\*

\* Launch the application.

\* @param <EditWindow>

\*/

public static void main(String[] args)

{

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

EditWindow frame = new EditWindow(-1,-1, null, null, null, null);

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the frame.

\*/

public EditWindow(int row, int itemIndex, Methods list, ArrayList<Item> items, MainMenu main, DefaultTableModel model)

{

setFont(new Font("Dialog", Font.PLAIN, 33));

setResizable(false);

setTitle("EDIT ITEMS");

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

setBounds(100, 100, 461, 420);

contentPane = new JPanel();

contentPane.setBackground(new Color(255, 255, 204));

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

contentPane.setLayout(null);

//JLabel lblItem = new JLabel("Items:");

//lblItem.setBounds(34, 21, 46, 14);

//contentPane.add(lblItem);

JLabel lblName = new JLabel("Name:");

lblName.setBounds(34, 39, 76, 14);

contentPane.add(lblName);

JLabel lblUsage = new JLabel("Weekly Usage:");

lblUsage.setBounds(34, 168, 112, 14);

contentPane.add(lblUsage);

JLabel lblComments = new JLabel("Comments:");

lblComments.setBounds(34, 253, 76, 14);

contentPane.add(lblComments);

JLabel lblStock = new JLabel("Stock:");

lblStock.setBounds(34, 124, 76, 14);

contentPane.add(lblStock);

JLabel lblCategory = new JLabel("Category:");

lblCategory.setBounds(34, 80, 92, 14);

contentPane.add(lblCategory);

//txtItems = new JComboBox<String>();

//for(int index = 0; index < items.size(); index++)

//{

// txtItems.addItem(items.get(index).getName());

//}

//txtItems.setSelectedIndex(-1);

//if(row >= 0)

// txtItems.setSelectedIndex(row);

//txtItems.setBounds(156, 18, 275, 20);

//contentPane.add(txtItems);

Item value = items.get(itemIndex);

txtName = new JTextField();

//if(row >= 0)

// txtName.setText(items.get(row).getName());

txtName.setText(value.getName());

txtName.setBounds(156, 36, 275, 20);

contentPane.add(txtName);

txtName.setColumns(10);

txtCategory = new JComboBox<String>();

txtCategory.addItem("Meat");

txtCategory.addItem("Vegetable");

txtCategory.addItem("Garnishes");

txtCategory.addItem("Seasoning");

txtCategory.addItem("Sauce");

txtCategory.addItem("Miscellaneous");

txtCategory.setSelectedIndex(-1);

String category = value.getCategory();

if(category.equals("Meat"))

txtCategory.setSelectedIndex(0);

else if(category.equals("Vegetable"))

txtCategory.setSelectedIndex(1);

else if(category.equals("Garnishes"))

txtCategory.setSelectedIndex(2);

else if(category.equals("Seasoning"))

txtCategory.setSelectedIndex(3);

else if(category.equals("Sauce"))

txtCategory.setSelectedIndex(4);

else //if(category.equals("Miscellaneous"))

txtCategory.setSelectedIndex(5);

//txtCategory.setSelectedIndex(txtCategory.getSelectedIndex());

txtCategory.setBounds(156, 74, 275, 20);

contentPane.add(txtCategory);

txtStock = new JTextField();

//if(row >= 0)

// txtStock.setText(Integer.toString(items.get(row).getStock()));

txtStock.setText(Integer.toString(value.getStock()));

txtStock.setColumns(10);

txtStock.setBounds(156, 121, 275, 20);

contentPane.add(txtStock);

txtUsage = new JTextField();

//if(row >= 0)

txtUsage.setText(Integer.toString(items.get(itemIndex).getUsage()));

txtUsage.setColumns(10);

txtUsage.setBounds(156, 165, 275, 20);

contentPane.add(txtUsage);

txtComments = new JTextField();

//if(row >= 0)

txtComments.setText(items.get(itemIndex).getComments());

txtComments.setHorizontalAlignment(SwingConstants.LEFT);

txtComments.setColumns(10);

txtComments.setBounds(156, 232, 275, 57);

contentPane.add(txtComments);

JLabel itemsError = new JLabel("");

itemsError.setForeground(Color.RED);

itemsError.setBounds(156, 11, 275, 14);

contentPane.add(itemsError);

JLabel nameError = new JLabel("");

nameError.setForeground(Color.RED);

nameError.setBounds(156, 56, 275, 14);

contentPane.add(nameError);

JLabel categoryError = new JLabel("");

categoryError.setForeground(Color.RED);

categoryError.setBounds(156, 96, 275, 14);

contentPane.add(categoryError);

JLabel stockError = new JLabel("");

stockError.setForeground(Color.RED);

stockError.setBounds(156, 140, 275, 14);

contentPane.add(stockError);

JLabel usageError = new JLabel("");

usageError.setForeground(Color.RED);

usageError.setBounds(156, 186, 275, 14);

contentPane.add(usageError);

//JLabel commentsError = new JLabel("");

//commentsError.setForeground(Color.RED);

//commentsError.setBounds(156, 318, 275, 14);

//contentPane.add(commentsError);

/\*

if( txtItems.getSelectedIndex() >= 0)

{

Item values = items.get(txtItems.getSelectedIndex());

txtName.setText(values.getName());

String category = values.getCategory();

if(category.equals("Meat"))

txtCategory.setSelectedIndex(0);

else if(category.equals("Vegetable"))

txtCategory.setSelectedIndex(1);

else if(category.equals("Garnishes"))

txtCategory.setSelectedIndex(2);

else if(category.equals("Seasoning"))

txtCategory.setSelectedIndex(3);

else if(category.equals("Sauce"))

txtCategory.setSelectedIndex(4);

else //if(category.equals("Miscellaneous"))

txtCategory.setSelectedIndex(5);

txtStock.setText(Integer.toString(values.getStock()));

txtUsage.setText(Integer.toString(values.getUsage()));

txtComments.setText(values.getComments());

}

\*/

JButton btnAdd = new JButton("EDIT");

btnAdd.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0)

{

//if( txtItems.getSelectedIndex() == -1)

//{

// itemsError.setText("Empty Slot: Please Select a Category");

//}

//else

//{

// itemsError.setText("");

if( txtName.getText().length() == 0)

{

nameError.setText("Empty Slot: Please Enter Name");

}

else

{

nameError.setText("");

if( txtCategory.getSelectedIndex() == -1)

{

categoryError.setText("Empty Slot: Please Select a Category");

}

else

{

categoryError.setText("");

try

{

Integer.parseInt(txtStock.getText());

if( Integer.parseInt(txtStock.getText()) < 0 || Integer.parseInt(txtStock.getText()) > 200)

{

stockError.setText("Please Enter in a Positive Integer");

}

else

{

stockError.setText("");

try

{

Integer.parseInt(txtUsage.getText());

if( Integer.parseInt(txtUsage.getText()) < 0 || Integer.parseInt(txtUsage.getText()) > 200)

{

usageError.setText("Please Enter in a Positive Integer");

}

else

{

usageError.setText("");

if( txtComments.getText().length() == 0)

{

int prompt = JOptionPane.showConfirmDialog(null, "Comments Section Blank, Leave Blank?", "COMMENT CHECKER", JOptionPane.YES\_NO\_OPTION);

if(prompt == JOptionPane.YES\_OPTION)

{

try

{

list.editItem(items.get(itemIndex).getName(),txtName.getText(), txtCategory.getSelectedItem().toString(),

Integer.parseInt(txtStock.getText()), Integer.parseInt(txtUsage.getText()), txtComments.getText());

list.modifyFile();

list.organizeList();

model.removeRow(row);

model.addRow(new Object[] {txtName.getText(), txtCategory.getSelectedItem().toString(), Integer.parseInt(txtStock.getText()),

Integer.parseInt(txtUsage.getText()), txtComments.getText() });

EditWindow.this.setVisible(false);

main.setVisible(true);

/\*

list.editItem(txtItems.getSelectedItem().toString(),txtName.getText(), txtCategory.getSelectedItem().toString(),

Integer.parseInt(txtStock.getText()), Integer.parseInt(txtUsage.getText()), txtComments.getText());

list.modifyFile();

list.organizeList();

model.removeRow(txtItems.getSelectedIndex());

model.addRow(new Object[] {txtName.getText(), txtCategory.getSelectedItem().toString(), Integer.parseInt(txtStock.getText()),

Integer.parseInt(txtUsage.getText()), txtComments.getText() });

//model.deleteRow(new Object[] {txtName.getText(), txtCategory.getSelectedItem().toString(), Integer.parseInt(txtStock.getText()),

// Integer.parseInt(txtUsage.getText()), txtComments.getText() });

EditWindow.this.setVisible(false);

main.setVisible(true);

\*/

}

catch(Exception e)

{

JOptionPane.showMessageDialog(null,"Error!!!");

}

}

}

else

{

try

{

list.editItem(items.get(itemIndex).getName(),txtName.getText(), txtCategory.getSelectedItem().toString(),

Integer.parseInt(txtStock.getText()), Integer.parseInt(txtUsage.getText()), txtComments.getText());

list.modifyFile();

list.organizeList();

model.removeRow(row);

model.addRow(new Object[] {txtName.getText(), txtCategory.getSelectedItem().toString(), Integer.parseInt(txtStock.getText()),

Integer.parseInt(txtUsage.getText()), txtComments.getText() });

EditWindow.this.setVisible(false);

main.setVisible(true);

}

catch(Exception e)

{

JOptionPane.showMessageDialog(null,"Error!!!");

}

}

}

}

catch (NumberFormatException e)

{

usageError.setText("Please Enter in a Valid Integer");

}

}

}

catch (NumberFormatException e)

{

stockError.setText("Please Enter in a Valid Integer");

}

}

// }

}

}

});

btnAdd.setBounds(50, 313, 150, 50);

contentPane.add(btnAdd);

JButton btnCancel = new JButton("CANCEL");

btnCancel.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e)

{

EditWindow.this.setVisible(false);

main.setVisible(true);

}

});

btnCancel.setBounds(250, 313, 150, 50);

contentPane.add(btnCancel);

}

}