package org.moflon.demo.leitnersbox;

import java.awt.BorderLayout;

import java.awt.Container;

import java.awt.event.ActionEvent;

import java.awt.event.KeyEvent;

import java.io.File;

import java.io.IOException;

import java.util.HashMap;

import java.util.Map;

import javax.swing.AbstractAction;

import javax.swing.ImageIcon;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JMenu;

import javax.swing.JMenuBar;

import javax.swing.JMenuItem;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

public class LeitnersBoxView {

private LeitnersBoxController controller;

private JMenu partitionsMenu;

private JMenu boxMenu;

private JFrame frame;

private Map<String, JMenu> submenus = new HashMap<>();

public LeitnersBoxView(LeitnersBoxController controller) {

controller.setView(this);

this.controller = controller;

}

public static void main(String[] args) {

javax.swing.SwingUtilities.invokeLater(new Runnable() {

public void run() {

new LeitnersBoxView(new LeitnersBoxController())

.createAndShowGUI();

}

});

}

private void createAndShowGUI() {

frame = new JFrame("Leitner's Box");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setJMenuBar(createMenuBar(frame));

frame.setContentPane(createContentPane());

frame.setSize(220, 220);

frame.setVisible(true);

controller.loadXmiFile();

controller.createTheBox();

}

public JMenu getPartitionsMenu() {

return partitionsMenu;

}

private JMenuBar createMenuBar(final JFrame frame) {

JMenuBar menuBar = new JMenuBar();

partitionsMenu = new JMenu("Partitions");

boxMenu = new JMenu("Box");

partitionsMenu.setMnemonic(KeyEvent.VK\_P);

boxMenu.setMnemonic(KeyEvent.VK\_P);

menuBar.add(partitionsMenu);

menuBar.add(boxMenu);

return menuBar;

}

public Container createContentPane() {

JPanel contentPane = new JPanel(new BorderLayout());

contentPane.setOpaque(true);

JLabel output = new JLabel(

createImageIcon("icons/noun\_project\_870.png"));

contentPane.add(output, BorderLayout.CENTER);

contentPane

.setToolTipText("Flash Cards designed by Rohan Gupta, "

+ "Memory designed by Anne Marie Nguyen, from The Noun Project.");

return contentPane;

}

private static ImageIcon createImageIcon(String filename) {

if (new File(filename).exists()) {

return new ImageIcon(filename);

} else {

System.err.println("Couldn't find file: " + filename);

return null;

}

}

void createBox(String name) {

JMenuItem growTheBox = new JMenuItem();

JMenuItem toString = new JMenuItem();

JMenu submenu = new JMenu(name);

submenus.put(name, submenu);

boxMenu.add(submenu);

submenus.get(name).add(growTheBox);

submenus.get(name).add(toString);

submenu.add(toString);

submenu.add(growTheBox);

setGrowAction(growTheBox);

setToString(toString,name);

toString.setText("give out String");

growTheBox.setText("add a new Partition?");

}

void createPartition(String name, int index) {

JMenuItem empty = new JMenuItem();

JMenu submenu = new JMenu(name);

submenu.setIcon(createImageIcon("icons/noun\_project\_4767.png"));

submenus.put(name, submenu);

partitionsMenu.add(submenu);

submenus.get(name).add(empty);

submenu.add(empty);

setEmptyAction(empty,index);

empty.setText("Empty Partition");

}

/\*\*void createCard(String submenu, final String cardName, int partIndex,

int cardIndex) {

JMenu subCard = new JMenu(cardName);

// Create Menu options for each action

JMenuItem cardRemove = new JMenuItem();

JMenuItem cardCheck = new JMenuItem();

submenus.get(submenu).add(subCard);

subCard.add(cardRemove);

subCard.add(cardCheck);

setRemoveCardAction(cardRemove, partIndex, cardIndex);

setCheckCardAction(cardCheck, partIndex, cardIndex);

cardRemove.setText("Remove Card");

cardCheck.setText("Check Card");

}

\*\*/

void createCard(String submenu, final String cardName, int partIndex,

int cardIndex) {

JMenu subCard = new JMenu(cardName);

// Create Menu options for each action

JMenuItem cardRemove = new JMenuItem();

JMenuItem cardCheck = new JMenuItem();

JMenuItem cardInvert = new JMenuItem();

submenus.get(submenu).add(subCard);

subCard.add(cardRemove);

subCard.add(cardCheck);

subCard.add(cardInvert);

setRemoveCardAction(cardRemove, partIndex, cardIndex);

setCheckCardAction(cardCheck, partIndex, cardIndex);

setInvertCardAction(cardInvert, partIndex, cardIndex);

cardRemove.setText("Remove Card");

cardCheck.setText("Check Card");

cardInvert.setText("Invert Card");

}

public void setInvertCardAction(JMenuItem card, final int partitionIndex,final int cardIndex) {

// TODO Auto-generated method stub

card.setAction(new AbstractAction() {

@Override

public void actionPerformed(ActionEvent e) {

controller.doActionInvertCard(partitionIndex, cardIndex);

}

});

}

@SuppressWarnings("serial")

public void setRemoveCardAction(JMenuItem card, final int partitionIndex,

final int cardIndex) {

card.setAction(new AbstractAction() {

@Override

public void actionPerformed(ActionEvent e) {

controller.doActionRemoveCard(partitionIndex, cardIndex);

}

});

}

public void setEmptyAction (JMenuItem partition,final int partitionIndex) {

partition.setAction(new AbstractAction(){

@Override

public void actionPerformed (ActionEvent e) {

controller.doActionEmpty(partitionIndex);

}

});

}

private void setGrowAction(JMenuItem growTheBox) {

// TODO Auto-generated method stub

growTheBox.setAction(new AbstractAction(){

public void actionPerformed (ActionEvent e) {

controller.doActionGrow();

}

});

}

@SuppressWarnings("serial")

public void setCheckCardAction(JMenuItem card, final int partIndex,

final int cardIndex) {

card.setAction(new AbstractAction() {

@Override

public void actionPerformed(ActionEvent e) {

controller.doActionCheckCard(partIndex, cardIndex);

}

});

}

private void setToString(JMenuItem toString, String name) {

// TODO Auto-generated method stub

toString.setAction(new AbstractAction() {

public void actionPerformed(ActionEvent e) {

controller.doActionToString(name);

}

});

}

public String getUserGuess() {

String guess = JOptionPane.showInputDialog("Your Guess:");

return guess;

}

public void refreshGUI() {

saveModel();

clearView();

controller.loadXmiFile();

}

public void saveModel() {

try {

controller.resource.save(null);

} catch (IOException e) {

e.printStackTrace();

}

}

public void clearView() {

submenus.clear();

partitionsMenu.removeAll();

}

}