/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

import javafx.application.Application;

import javafx.scene.Scene;

import javafx.scene.layout.StackPane;

import javafx.scene.layout.VBox;

import javafx.stage.Stage;

import view.MainView;

public class Main extends Application {

@Override

public void start(Stage primaryStage) {

StackPane root = new StackPane();

VBox imageBox = new VBox();

MainView view = new MainView(primaryStage, imageBox);

root.getChildren().setAll(view, imageBox);

Scene scene = new Scene(root);

primaryStage.setTitle("Library");

primaryStage.setScene(scene);

primaryStage.sizeToScene();

primaryStage.show();

}

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

launch(args);

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package controller;

import externalfile.SaveAndLoad;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import javafx.stage.Stage;

import model.Book;

import model.CollectionOfBooks;

import view.AddBookView;

import view.AlertView;

import view.SaveAnimationView;

import view.CenterTableView;

import view.ExitVBoxView;

import view.FileChooserView;

public class Controller {

private CollectionOfBooks library;

private AddBookView addBookView;

private CenterTableView centerTableView;

private FileChooserView fileChooserView;

private SaveAndLoad saveAndLoad;

private ExitVBoxView exitView;

private Stage stage;

private SaveAnimationView saveAniView;

private AlertView alertView;

public Controller(CollectionOfBooks library,

CenterTableView centerTableView, FileChooserView fileChooserView,

Stage stage, SaveAnimationView saveAniView) {

this.library = library;

this.centerTableView = centerTableView;

this.fileChooserView = fileChooserView;

this.stage = stage;

this.saveAniView = saveAniView;

saveAndLoad = new SaveAndLoad();

alertView = new AlertView();

}

public void addBook() {

addBookView = new AddBookView(this);

}

public boolean handleInput() {

addBookView.setBlackLabels();

List<String> tmp = addBookView.getInfo();

Boolean error = false;

Double price = 0.0;

int edition = 0;

for (int i = 0; i < 5; i++) {

if (tmp.get(i).length() <= 0) {

addBookView.setRedLabels(i);

error = true;

}

}

try {

edition = Integer.parseInt(tmp.get(2));

if (edition < 0) {

addBookView.setRedLabels(2);

error = true;

}

}

catch(NumberFormatException e) {

addBookView.setRedLabels(2);

error = true;

}

try {

price = Double.parseDouble(tmp.get(3));

if (price < 0) {

addBookView.setRedLabels(3);

error = true;

}

}

catch(NumberFormatException e) {

addBookView.setRedLabels(3);

error = true;

}

if (error) {

alertView.showAlert("Make sure you fill all fields!\n"

+ "Make sure Edition and Price are Positive numbers!");

return false;

}

else {

String tmpS = tmp.get(4);

String author = "";

ArrayList<String> authors = new ArrayList();

for (int i = 0; i < tmpS.length(); i++) {

if (tmpS.charAt(i) != ',') {

author += tmpS.charAt(i);

}

else {

authors.add(author.trim());

author = "";

}

}

if (author.length() != 0)

authors.add(author.trim());

System.out.println(author.length());

library.addBook(new Book(tmp.get(0), tmp.get(1), edition, price, authors));

addBookView.exitStage();

return true;

}

}

public void handleAddBookCancel() {

addBookView.exitStage();

}

public void removeBook() {

ArrayList<Book> tmp = centerTableView.removeBook();

for (Book b : tmp) {

library.removeBook(b);

}

}

public void searchBook(String searchedFor, String searched) {

if ("Title".equals(searchedFor))

centerTableView.setSearchedList(library.searchByTitle(searched));

else if("ISBN".equals(searchedFor))

centerTableView.setSearchedList(library.searchByIsbn(searched));

else if("Author".equals(searchedFor))

centerTableView.setSearchedList(library.searchByAuthor(searched));

else

centerTableView.setSearchedList(library.searchByTitle(searched));

}

public void refresh() {

centerTableView.refresh();

}

public void exitProgram(){

exitView = new ExitVBoxView(this);

}

public void closeWithSaving() throws IOException {

saveToFile();

exitView.close();

stage.close();

}

public void closeWithoutSaving() {

exitView.close();

stage.close();

}

public void closeCanceled() {

exitView.close();

}

public void saveToFile() throws IOException {

String path = fileChooserView.saveToFile();

if (path != null) {

if (saveAndLoad.objectOutput(path, library))

saveAniView.startAnimation();

else

alertView.showAlert("File did not save!");

}

else

alertView.showAlert("File did not save!");

}

public void saveAsToFile() throws IOException {

String path = fileChooserView.saveAsToFile();

if (path != null) {

if (saveAndLoad.objectOutput(path, library))

saveAniView.startAnimation();

else

alertView.showAlert("File did not save!");

}

else

alertView.showAlert("File did not save!");

}

public void loadFromFile() throws IOException {

String path = fileChooserView.loadFromFile();

if (path != null) {

if (saveAndLoad.objectInput(path) != null) {

library.setBooks(saveAndLoad.objectInput(path));

centerTableView.refresh();

}

else

alertView.showAlert("File did not load!");

}

else

alertView.showAlert("File did not load!");

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package externalfile;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import model.Book;

import model.CollectionOfBooks;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.util.ArrayList;

public class SaveAndLoad {

public SaveAndLoad() {

}

public ArrayList<Book> objectInput(String path) throws IOException {

ObjectInputStream inputFile = null;

try {

inputFile = new ObjectInputStream(new FileInputStream(path));

return readFromFile(inputFile);

}

catch (Exception e) {

return null;

}

finally {

if (inputFile != null) {

inputFile.close();

}

}

}

public boolean objectOutput(String path, CollectionOfBooks library) throws IOException {

ObjectOutputStream outputFile = null;

try {

outputFile = new ObjectOutputStream(new FileOutputStream(path));

saveToFile(outputFile, library);

return true;

}

catch (Exception e) {

return false;

}

finally {

if (outputFile != null) {

outputFile.close();

}

}

}

public ArrayList<Book> readFromFile(ObjectInputStream input) throws IOException, ClassNotFoundException {

return (ArrayList<Book>) input.readObject();

}

public void saveToFile(ObjectOutputStream output, CollectionOfBooks library) throws IOException {

output.writeObject(library.getList());

}

}

package model;

import java.io.Serializable;

import java.util.Observable;

/\*\*

\* An <code>Author</code> has a <code>name</code>.

\* <code>Author</code> extends <code>Observable</code> in order to notify obsrvers of changes.

\* <code>Author</code> implements <code>Serializable</code> in order to save object in a file.

\* <code>Author</code> implements <code>Comparable</code> in order to compare itself with itself

\* to other <code>Author</code>s.

\*

\* @author Niklas Ålander

\* @version 1.1

\*/

public class Author extends Observable implements Serializable, Comparable<Author> {

private String name;

/\*\*

\* Creates a new <code>Author</code> object.

\* @param name The name of the new <code>Author</code>.

\*/

public Author(String name) {

this.name = name;

}

/\*\*

\* Returns the <code>name</code> of the <code>Author</code>.

\* @return Returns the <code>name</code> of the <code>Author</code>.

\*/

public String getName() {

return name;

}

/\*\*

\* Changes the authors <code>name</code>.

\* @param name The authors new <code>name</code>.

\*/

public void setName(String name) {

this.name = name;

notifyAllObservers();

}

/\*\*

\* Compares the name of two different <code>Author</code>s.

\* @param other The <code>other</code> <code>Author</code>s name.

\* @return Returns 0 if the authors names are the same,

\* return a negative number if <code>this</code> authors <code>name</code>

\* is less than <code>other</code> authors <code>name</code>,

\* return a positive number if <code>this</code> authors <code>name</code>

\* is greater than <code>other</code> authors <code>name</code>.

\*/

@Override

public int compareTo(Author other) {

String tmp = this.name;

String otherTmp = other.name;

return tmp.compareTo(otherTmp);

}

/\*\*

\* Notifies observers that a change has happened.

\*/

public void notifyAllObservers() {

this.setChanged();

this.notifyObservers();

}

/\*\*

\* Creates a text <code>String</code> of the authors <code>name</code>.

\* @return Returns a text <code>String</code> of the authors <code>name</code>.

\*/

public String toString() {

String text = new String(name);

return text;

}

}

package model;

import java.util.Comparator;

import java.util.Collections;

import java.util.ArrayList;

/\*\*

\* authorComp takes the first <code>Author</code> of two <code>Book</code>s and compares them.

\* authorComp implements <code>Comparator</code> in order to compare <code>Author</code>.

\*

\* @author Niklas Ålander

\* @version 1.0

\*/

public class AuthorSort implements Comparator<Book> {

/\*\*

\* Compares the first <code>Author</code> of two different <code>Book</code>s.

\* @param b1 <code>Book</code> 1 to be compared.

\* @param b2 <code>Book</code> 2 to be compared.

\* @return Returns 0 if the <code>Author</code>s are the same,

\* return a negative number if <code>b1</code>s <code>Author</code>

\* is less than <code>b2</code>s <code>Author</code>,

\* return a positive number if <code>b1</code>s <code>Author</code>

\* is greater than <code>b2</code>s <code>Author</code>,

\*/

@Override

public int compare(Book b1, Book b2) {

ArrayList<Author> a1 = b1.getsAuthors();

ArrayList<Author> a2 = b2.getsAuthors();

Collections.sort(a1, new AuthorSort2());

Collections.sort(a2, new AuthorSort2());

return a1.get(0).getName().compareTo(a2.get(0).getName());

}

}

package model;

import java.util.Comparator;

import java.util.LinkedList;

/\*\*

\* authorSort sorts <code>Author</code>s of a <code>Book</code> in alphabetical order.

\* authorSort implements <code>Comparator</code> in order to sort <code>Author</code>s.

\*

\* @author Niklas Ålander

\* @version 1.0

\*/

public class AuthorSort2 implements Comparator<Author> {

/\*\*

\* Sorts <code>Author</code>s of a <code>Book</code> in alphabetical order.

\* @param a1 <code>Author</code> 1 to be sorted.

\* @param a2 <code>Author</code> 2 to be sorted.

\* @return Returns 0 if the <code>Author</code>s <code>name</code>s are the same,

\* return a negative number if <code>a1</code>s <code>name</code>

\* is less than <code>a2</code>s <code>name</code>,

\* return a positive number if <code>a1</code>s <code>name</code>

\* is greater than <code>a2</code>s <code>name</code>,

\*/

@Override

public int compare(Author a1, Author a2) {

String name1 = a1.getName();

String name2 = a2.getName();

return name1.compareTo(name2);

}

}

package model;

import java.util.ArrayList;

import java.io.Serializable;

import java.util.Observable;

/\*\*

\* A <code>Book</code> has a <code>title</code>, <code>isbn</code>, <code>edition</code>,

\* <code>price</code> and a List of <code>Author</code>s.

\* <code>Book</code> implements <code>Serializable</code> in order to save object in a file.

\*

\* @author Niklas Ålander

\* @version 1.1

\*/

public class Book extends Observable implements Serializable {

private String title;

private String isbn;

private int edition;

private double price;

private ArrayList<Author> authors;

/\*\*

\* Creates a new <code>Book</code>.

\* @param title The title.

\* @param isbn The isbn.

\* @param edition The edition.

\* @param price The price.

\* @param authors The list of all authors names.

\*/

public Book(String title, String isbn, int edition, double price, ArrayList<String> authors) {

this.title = title;

this.isbn = isbn;

this.edition = edition;

this.price = price;

this.authors = new ArrayList<Author>();

for (String s : authors)

this.authors.add(new Author(s));

}

/\*\*

\* Returns the <code>title</code> of the <code>Book</code>.

\* @return Returns the <code>title</code> of the <code>Book</code>.

\*/

public String getTitle() {

return title;

}

/\*\*

\* Changes the <code>Book</code> <code>title</code>.

\* @param title The <code>Book/code> new <code>title</code>.

\*/

public void setTitle(String title) {

this.title = title;

notifyAllObservers();

}

/\*\*

\* Returns the <code>isbn</code> of the <code>Book</code>.

\* @return Returns the <code>isbn</code> of the <code>Book</code>.

\*/

public String getIsbn() {

return isbn;

}

/\*\*

\* Changes the <code>Book</code> <code>isbn</code>.

\* @param isbn The <code>Book/code> new <code>isbn</code>.

\*/

public void setIsbn(String isbn) {

this.isbn = isbn;

notifyAllObservers();

}

/\*\*

\* Returns the <code>edition</code> of the <code>Book</code>.

\* @return Returns the <code>edition</code> of the <code>Book</code>.

\*/

public int getEdition() {

return edition;

}

/\*\*

\* Changes the <code>Book</code> <code>edition</code>.

\* @param edition The <code>Book/code> new <code>edition</code>.

\*/

public void setEdition(int edition) {

this.edition = edition;

notifyAllObservers();

}

/\*\*

\* Returns the <code>price</code> of the <code>Book</code>.

\* @return Returns the <code>price</code> of the <code>Book</code>.

\*/

public double getPrice() {

return price;

}

/\*\*

\* Changes the <code>Book</code> <code>price</code>.

\* @param price The <code>Book/code> new <code>price</code>.

\*/

public void setPrice(double price) {

this.price = price;

notifyAllObservers();

}

/\*\*

\* Creates a <code>clone</code> of the list of <code>Author</code>s

\* of the <code>Book</code> to be returned.

\* @return Creates a <code>clone</code> of the list of

\* <code>Author</code>s of the <code>Book</code> to be returned.

\*/

public ArrayList<Author> getsAuthors() {

return (ArrayList<Author>) authors.clone();

}

/\*\*

\* Creates a <code>String</code> of the list of <code>Author</code>s

\* of the <code>Book</code> to be returned.

\* @return Creates a <code>String</code> of the list of

\* <code>Author</code>s of the <code>Book</code> to be returned.

\*/

public String getAuthors() {

String tmp = new String("");

for (Author a : authors) {

tmp += a.getName() + ", ";

}

return tmp;

}

/\*\*

\* Creates a new <code>Author</code> and adds it to the list.

\* @param name The name of the new <code>Author</code>.

\*/

public void addAuthor(String name) {

authors.add(new Author(name));

notifyAllObservers();

}

/\*\*

\* Creates a new <code>Author</code> and adds it to the list.

\* @param author The new <code>Author</code>.

\*/

public void addAuthor(Author author) {

authors.add(author);

}

/\*\*

\* Notifies observers that a change has happened.

\*/

public void notifyAllObservers() {

this.setChanged();

this.notifyObservers();

}

/\*\*

\* Creates a text <code>String</code> with information

\* about the <code>Book</code>.

\* @return Returns a text <code>String</code> with information

\* about the <code>Book</code>.

\*/

public String toString() {

String text = new String("");

text += "Title: " + title;

text += "\nISBN: " + isbn;

text += "\nEdition: " + edition;

text += "\nPrice: " + price;

text += "\nAuthor(s): ";

for (Author a : authors) {

text += a.toString() + ", ";

}

return text;

}

}

package model;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Observable;

/\*\*

\* A CollectionOfBooks contains a ArrayList of <code>Book</code> objects.

\* The list can be manipulated by adding and removing <code>Book</code> objects.

\*

\* @author Niklas Ålander

\* @version 1.1

\*/

public class CollectionOfBooks extends Observable {

//private ArrayList<Book> library;

private ArrayList<Book> library;

/\*\*

\* Constructs a new ArrayList of <code>Book</code> objects.

\*/

public CollectionOfBooks() {

library = new ArrayList<Book>();

}

/\*\*

\* Constructs a <code>ArrayList</code> of <code>Book</code> objects.

\* @param library The new list of <code>Book</code>.

\*/

public CollectionOfBooks(ArrayList<Book> library) {

this.library = library;

}

/\*\*

\* Changes the <code>CollectionOfBooks</code> <code>books</code>.

\* @param library The new list of <code>Book</code>.

\*/

public void setBooks(ArrayList<Book> library) {

this.library = library;

notifyAllObservers();

}

/\*\*

\* Return the list of type <code>Book</code>.

\* @return Return the list of type <code>Book</code>.

\*/

public ArrayList<Book> getList() {

return (ArrayList<Book>) library;

}

/\*\*

\* Return the size of the list.

\* @return Returns the size of list.

\*/

public int getSize() {

return library.size();

}

/\*\*

\* Adds a new <code>Book</code> to the list.

\* @param book The new <code>Book</code> to add to the list.

\*/

public void addBook(Book book) {

library.add(book);

notifyAllObservers();

}

/\*\*

\* Removes a book from the list if given index is within the list and returns

\* <code>true</code> or <code>false</code> whether it was successful.

\* @param index The place in the list which specifies which book to remove.

\* @return Return <code>true</code> if the book was successfully removed,

\* otherwise return <code>false</code>.

\*/

public boolean removeBook(int index) {

if (index >= 0 && index < library.size()) {

library.remove(index);

notifyAllObservers();

return true;

}

else

return false;

}

/\*\*

\* Removes a book from the list if given <code>Book</code> is

\* within the list and returns <code>true</code>.

\* @param book The Book that is to be removed.

\* @return Return <code>true</code> when the book has been removed.

\*/

public boolean removeBook(Book book) {

library.remove(book);

notifyAllObservers();

return true;

}

/\*\*

\* Removes a book from the list if given index is within the list and returns

\* removed book or <code>null</code> is it was unsuccessful.

\* @param index The place in the list which specifies which book to remove.

\* @return If the book was successfully removed, return the book,

\* otherwise return <code>null</code>.

\*/

public Book removeAndGetBook(int index) {

if (index >= 0 && index < library.size()) {

notifyAllObservers();

return library.remove(index);

}

else

return null;

}

/\*\*

\* Changes the name of an <code>Author</code>.

\* @param author The <code>Author</code> whos name is to be changed.

\* @param name The new name of specified <code>Author</code>.

\*/

public void changeAuthorName(Author author, String name) {

author.setName(name);

notifyAllObservers();

}

/\*\*

\* Adds an <code>Author</code> to a <code>Book</code> object.

\* @param index The index in the list of library that will gain an <code>Author</code>.

\* @param name The name of the new <code>Author</code>.

\* @return Return <code>true</code> if the <code>Author</code> was successfully added,

\* otherwise return <code>false</code>.

\*/

public boolean addAuthor(int index, String name) {

if (index >= 0 && index < library.size()) {

Author tmpAuthor = findAuthor(name);

if (tmpAuthor != null) {

library.get(index).addAuthor(tmpAuthor);

notifyAllObservers();

return true;

}

else {

library.get(index).addAuthor(name);

notifyAllObservers();

return true;

}

}

else

return false;

}

/\*\*

\* Searches all library to find an <code>Author</code> to return.

\* @param name The name of the <code>Author</code> to be found.

\* @return If an <code>Author</code> is found return the <code>Author</code>,

\* otherwise return <code>null</code>.

\*/

public Author findAuthor(String name) {

Author tmpAuthor = new Author(name);

for (Book b : library) {

ArrayList<Author> tmpList = b.getsAuthors();

for (Author a : tmpList) {

if (a.compareTo(tmpAuthor) == 0) {

return a;

}

}

}

return null;

}

/\*\*

\* Searches for library by its <code>isbn</code>.

\* If a book's <code>isbn</code> contains the search word, the book

\* will be added to a list which will be returned.

\* @param isbn The <code>isbn</code> that is being searched.

\* @return Returns the list of library that had matching isbns.

\*/

public ArrayList<Book> searchByIsbn(String isbn) {

ArrayList<Book> tmpBookList = new ArrayList<Book>();

CharSequence tmpIsbn = isbn.toUpperCase();

for (Book b : library) {

if (b.getIsbn().toUpperCase().contains(tmpIsbn))

tmpBookList.add(b);

}

Collections.sort(tmpBookList, new IsbnSort());

return tmpBookList;

}

/\*\*

\* Searches for library by its <code>title</code>.

\* If a book's <code>title</code> contains the search word, the book

\* will be added to a list which will be returned.

\* @param title The <code>title</code> that is being searched.

\* @return Returns the list of library that had matching titles.

\*/

public ArrayList<Book> searchByTitle(String title) {

ArrayList<Book> tmpBookList = new ArrayList<Book>();

CharSequence tmpTitle = title.toUpperCase();

for (Book b : library) {

if (b.getTitle().toUpperCase().contains(tmpTitle))

tmpBookList.add(b);

}

Collections.sort(tmpBookList, new TitleSort());

return tmpBookList;

}

/\*\*

\* Searches for library by its <code>Author</code>.

\* If a book's <code>Author</code> contains the search word, the book

\* will be added to a list which will be returned.

\* @param name The <code>Author</code> that is being searched.

\* @return Returns the list of library that had matching Authors.

\*/

public ArrayList<Book> searchByAuthor(String name) {

ArrayList<Book> tmpBookList = new ArrayList<Book>();

CharSequence tmpName = name.toUpperCase();

Author tmpAuthor = new Author(name);

for (Book b : library) {

ArrayList<Author> tmpList = b.getsAuthors();

for (Author a : tmpList) {

if (a.getName().toUpperCase().contains(tmpName)) {

tmpBookList.add(b);

break;

}

}

}

Collections.sort(tmpBookList, new AuthorSort());

return tmpBookList;

}

/\*\*

\* Notifies observers that a change has happened.

\*/

public void notifyAllObservers() {

this.setChanged();

this.notifyObservers();

}

/\*\*

\* Creates a text <code>String</code> with information of every

\* <code>Book</code> is put together.

\* @return Returns a text <code>String</code> with information

\* of every <code>Book</code> in the list.

\*/

public String toString() {

String text = new String("\n");

int i = 0;

for (Book b : library) {

text += "\nIndex: " + i + "\n" + b.toString() + "\n";

i += 1;

}

return text;

}

}

package model;

import java.util.Comparator;

/\*\*

\* isbnComp takes the isbn of two <code>Book</code>s and compares them.

\* isbnComp implements <code>Comparator</code> in order to compare <code>isbn</code>.

\*

\* @author Niklas Ålander

\* @version 1.0

\*/

public class IsbnSort implements Comparator<Book> {

/\*\*

\* Compares the isbn of two different <code>Book</code>s.

\* @param b1 <code>Book</code> 1 to be compared.

\* @param b2 <code>Book</code> 2 to be compared.

\* @return Returns 0 if the <code>isbn</code>s are the same,

\* return a negative number if <code>b1</code>s <code>isbn</code>

\* is less than <code>b2</code>s <code>isbn</code>,

\* return a positive number if <code>b1</code>s <code>isbn</code>

\* is greater than <code>b2</code>s <code>isbn</code>,

\*/

@Override

public int compare(Book b1, Book b2) {

String isbn1 = b1.getIsbn();

String isbn2 = b2.getIsbn();

return isbn1.compareTo(isbn2);

}

}package model;

import java.util.Comparator;

/\*\*

\* titleComp takes the title of two <code>Book</code>s and compares them.

\* titleComp implements <code>Comparator</code> in order to compare <code>title</code>.

\*

\* @author Niklas Ålander

\* @version 1.0

\*/

public class TitleSort implements Comparator<Book> {

/\*\*

\* Compares the title of two different <code>Book</code>s.

\* @param b1 <code>Book</code> 1 to be compared.

\* @param b2 <code>Book</code> 2 to be compared.

\* @return Returns 0 if the <code>title</code>s are the same,

\* return a negative number if <code>b1</code>s <code>title</code>

\* is less than <code>b2</code>s <code>title</code>,

\* return a positive number if <code>b1</code>s <code>title</code>

\* is greater than <code>b2</code>s <code>title</code>,

\*/

@Override

public int compare(Book b1, Book b2) {

String title1 = b1.getTitle();

String title2 = b2.getTitle();

return title1.compareTo(title2);

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package view;

import controller.Controller;

import java.util.ArrayList;

import java.util.List;

import javafx.event.ActionEvent;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

import javafx.scene.control.TextField;

import javafx.scene.control.Tooltip;

import javafx.scene.layout.BorderPane;

import javafx.scene.layout.GridPane;

import javafx.scene.layout.HBox;

import javafx.scene.paint.Color;

import javafx.stage.Modality;

import javafx.stage.Stage;

public class AddBookView extends Stage{

private Controller controller;

private Label titleL;

private Label isbnL;

private Label editionL;

private Label priceL;

private Label authorL;

private TextField title;

private TextField isbn;

private TextField edition;

private TextField price;

private TextField author;

private Button addBook;

private Button cancel;

private GridPane grid;

private HBox hBox;

private BorderPane border;

public AddBookView(Controller controller) {

this.controller = controller;

initView();

}

private void initView() {

// create new labels, textfields and buttons

initFields();

clearFields();

// create new gridpane

initGridPane();

// create new hbox

initHbox();

// create new borderpane

initBorderPane();

//event handerls for this stage

addEventHandlers();

// create the new stage

initStage();

}

private void initFields() {

titleL = new Label("Title:");

isbnL = new Label("Isbn:");

editionL = new Label("Edition:");

priceL = new Label("Price:");

authorL = new Label("Authors:");

Tooltip authorTip= new Tooltip("Use ',' to separate authors!");

authorL.setTooltip(authorTip);

setBlackLabels();

title = new TextField();

isbn = new TextField();

edition = new TextField();

price = new TextField();

author = new TextField();

author.setTooltip(authorTip);

title.setPromptText("Type title here");

isbn.setPromptText("Type isbn here");

edition.setPromptText("Type edition here");

price.setPromptText("Type price here");

author.setPromptText("Type authors here");

addBook = new Button("Add Book");

cancel = new Button("Cancel");

}

public void clearFields() {

title.clear();

isbn.clear();

edition.clear();

price.clear();

author.clear();

}

private void initGridPane() {

grid = new GridPane();

grid.setHgap(10);

grid.setVgap(10);

grid.setPadding(new Insets(5, 20, 5, 20));

grid.add(titleL, 0, 0);

grid.add(isbnL, 0, 1);

grid.add(editionL, 0, 2);

grid.add(priceL, 0, 3);

grid.add(authorL, 0, 4);

grid.add(title, 1, 0);

grid.add(isbn, 1, 1);

grid.add(edition, 1, 2);

grid.add(price, 1, 3);

grid.add(author, 1, 4);

}

private void initHbox() {

hBox = new HBox(10);

hBox.setAlignment(Pos.CENTER);

hBox.setPadding(new Insets(10, 10, 10, 10));

hBox.getChildren().addAll(addBook, cancel);

}

private void initBorderPane() {

border = new BorderPane();

border.setCenter(grid);

border.setBottom(hBox);

}

private void initStage() {

Scene scene = new Scene(border);

// user can't interact with any other stage

this.initModality(Modality.APPLICATION\_MODAL);

this.setResizable(true);

this.setTitle("Add a new Book");

this.setScene(scene);

this.show();

}

public void setBlackLabels() {

titleL.setTextFill(Color.BLACK);

isbnL.setTextFill(Color.BLACK);

editionL.setTextFill(Color.BLACK);

priceL.setTextFill(Color.BLACK);

authorL.setTextFill(Color.BLACK);

}

public void setRedLabels(int index) {

if (index == 0)

titleL.setTextFill(Color.RED);

else if (index == 1)

isbnL.setTextFill(Color.RED);

else if (index == 2)

editionL.setTextFill(Color.RED);

else if (index == 3)

priceL.setTextFill(Color.RED);

else if (index == 4)

authorL.setTextFill(Color.RED);

}

public List<String> getInfo() {

List<String> tmp = new ArrayList();

tmp.add(title.getText().trim());

tmp.add(isbn.getText().trim());

tmp.add(edition.getText().trim());

tmp.add(price.getText().trim());

tmp.add(author.getText().trim());

return tmp;

}

public void exitStage() {

clearFields();

this.close();

}

private void addEventHandlers() {

addBook.setOnAction((ActionEvent event) -> {

controller.handleInput();

});

cancel.setOnAction((ActionEvent event) -> {

controller.handleAddBookCancel();

});

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package view;

import javafx.scene.control.Alert;

public class AlertView {

private Alert alert = new Alert(Alert.AlertType.INFORMATION);

public void showAlert(String message) {

alert.setHeaderText("");

alert.setTitle("Alert!");

alert.setContentText("Make sure you fill all fields!\n"

+ "Make sure Edition and Price are Positive numbers!");

alert.show();

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package view;

import model.Author;

import java.util.ArrayList;

import java.util.Observable;

import java.util.Observer;

import javafx.collections.FXCollections;

import model.Book;

import model.CollectionOfBooks;

import javafx.collections.ObservableList;

import javafx.scene.control.TableColumn;

import javafx.scene.control.TableView;

import javafx.scene.control.cell.PropertyValueFactory;

public class CenterTableView extends TableView implements Observer {

private CollectionOfBooks library;

private ObservableList<Book> observBooks;

private TableColumn title;

private TableColumn isbn;

private TableColumn edition;

private TableColumn price;

private TableColumn author;

public CenterTableView (CollectionOfBooks library) {

this.library = library;

initView();

}

public ArrayList<Book> removeBook() {

ObservableList<Book> librarySelected;

librarySelected = this.getSelectionModel().getSelectedItems();

ArrayList<Book> tmp = new ArrayList();

for (Book b : librarySelected) {

tmp.add(b);

}

return tmp;

}

private void initView() {

this.setEditable(true);

title = new TableColumn("Title");

title.setMinWidth(100);

isbn = new TableColumn("Isbn");

isbn.setMinWidth(80);

edition = new TableColumn("Edition");

edition.setMinWidth(80);

price = new TableColumn("Price");

price.setMinWidth(80);

author = new TableColumn("Author");

author.setMinWidth(128);

this.getColumns().addAll(title, isbn, edition, price, author);

title.setCellValueFactory(

new PropertyValueFactory<Book, String>("title"));

isbn.setCellValueFactory(

new PropertyValueFactory<Book, String>("isbn"));

edition.setCellValueFactory(

new PropertyValueFactory<Book, Integer>("edition"));

price.setCellValueFactory(

new PropertyValueFactory<Book, Double>("price"));

author.setCellValueFactory(

new PropertyValueFactory<Book, ArrayList<Author>>("authors"));

observBooks = FXCollections.observableArrayList(library.getList());

refresh();

}

public void setSearchedList(ArrayList<Book> tmpBooks) {

ObservableList<Book> tmpObvBooks;

tmpObvBooks = FXCollections.observableArrayList(tmpBooks);

this.setItems(tmpObvBooks);

}

public void refresh() {

observBooks.removeAll(observBooks);

observBooks.addAll(library.getList());

this.setItems(observBooks);

}

@Override

public void update(Observable o, Object o1) {

refresh();

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package view;

import controller.Controller;

import java.io.IOException;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.control.Button;

import javafx.scene.layout.HBox;

public class ExitHBoxView extends HBox{

private Button yesButton;

private Button noButton;

private Button cancelButton;

public ExitHBoxView(Controller controller){

super(10);

initView(controller);

}

private void initView(Controller controller) {

this.setAlignment(Pos.CENTER);

this.setPadding(new Insets(10, 10, 10, 10));

yesButton = new Button("Save");

noButton = new Button("Don't save");

cancelButton = new Button("Cancel");

yesButton.setPrefWidth(90);

noButton.setPrefWidth(90);

cancelButton.setPrefWidth(90);

this.getChildren().addAll(yesButton, noButton, cancelButton);

addHandlers(controller);

}

private void addHandlers(Controller controller) {

yesButton.setOnAction(event ->{

try {

controller.closeWithSaving();

}

catch (IOException e) {

}

});

noButton.setOnAction(event ->{

controller.closeWithoutSaving();

});

cancelButton.setOnAction(event ->{

controller.closeCanceled();

});

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package view;

import controller.Controller;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.Label;

import javafx.stage.Modality;

import javafx.stage.Stage;

import javafx.scene.layout.VBox;

public class ExitVBoxView extends VBox {

private Stage stage;

private String message;

private Label label;

private ExitHBoxView exitHBox;

public ExitVBoxView(Controller controller){

super(10);

this.stage = new Stage();

this.message = "Do you wish to save before you Quit?";

initView(controller);

}

private void initView(Controller controller) {

stage.initModality(Modality.APPLICATION\_MODAL);

stage.setTitle("Save");

System.out.println(message);

label = new Label();

label.setText(message);

exitHBox = new ExitHBoxView(controller);

this.getChildren().addAll(label,exitHBox);

this.setAlignment(Pos.CENTER);

this.setPadding(new Insets(5, 10, 5, 10));

Scene scene = new Scene(this);

stage.setTitle("Save before exit?");

stage.setScene(scene);

stage.sizeToScene();

stage.show();

}

public void close() {

stage.close();

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package view;

import java.io.File;

import javafx.stage.FileChooser;

import javafx.stage.Stage;

public class FileChooserView {

private Stage stage;

private String path;

private FileChooser fileChooser;

public FileChooserView(Stage stage) {

this.stage = stage;

path = "";

initView();

}

private void initView() {

fileChooser = new FileChooser();

fileChooser.setTitle("Open a file");

}

public String saveToFile() {

if (path.length() > 0)

return path;

else

return saveAsToFile();

}

public String saveAsToFile() {

FileChooser fileChooser = new FileChooser();

fileChooser.setTitle("Open a file");

fileChooser.getExtensionFilters().addAll(

new FileChooser.ExtensionFilter("SER", "\*.ser"),

new FileChooser.ExtensionFilter("All Files", "\*.\*"));

try {

File file = fileChooser.showSaveDialog(stage);

path = file.getPath();

return path;

}

catch (Exception e) {

path = "";

return null;

}

}

public String loadFromFile() {

FileChooser fileChooser = new FileChooser();

fileChooser.setTitle("Open a file");

fileChooser.getExtensionFilters().addAll(

new FileChooser.ExtensionFilter("SER", "\*.ser"),

new FileChooser.ExtensionFilter("All Files", "\*.\*"));

try {

File file = fileChooser.showOpenDialog(stage);

path = file.getPath();

return path;

}

catch (Exception e) {

path = "";

return null;

}

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package view;

import controller.Controller;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import model.CollectionOfBooks;

import javafx.scene.layout.Priority;

import javafx.scene.layout.VBox;

import javafx.stage.Stage;

public class MainView extends VBox {

private CollectionOfBooks library;

private Stage stage;

private Controller controller;

public MainView(Stage stage, VBox imageBox) {

this.stage = stage;

initView(imageBox);

}

private void initView(VBox imageBox) {

library = new CollectionOfBooks();

SaveAnimationView saveAniView = new SaveAnimationView(imageBox);

CenterTableView centerTable = new CenterTableView(library);

FileChooserView fileChooser = new FileChooserView(stage);

this.controller = new Controller(library, centerTable,

fileChooser, stage, saveAniView);

TopHboxView topHbox = new TopHboxView(controller);

SearchFieldView searchField = new SearchFieldView(controller);

MenuFieldView menuField = new MenuFieldView(controller);

library.addObserver(centerTable);

this.setPrefSize(470, 420);

this.setAlignment(Pos.CENTER);

this.setPadding(new Insets(0, 0, 0, 0));

this.getChildren().addAll(menuField, topHbox, searchField, centerTable);

this.setVgrow(centerTable, Priority.ALWAYS);

saveBeforeQuit();

}

private void saveBeforeQuit(){

stage.setOnCloseRequest(event -> {

event.consume();

controller.exitProgram();

});

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package view;

import controller.Controller;

import java.io.IOException;

import javafx.event.ActionEvent;

import javafx.scene.control.Menu;

import javafx.scene.control.MenuBar;

import javafx.scene.control.MenuItem;

public class MenuFieldView extends MenuBar{

private Controller controller;

private MenuItem save;

private MenuItem saveAs;

private MenuItem load;

private MenuItem exit;

public MenuFieldView(Controller controller) {

super();

this.controller = controller;

initView();

}

private void initView() {

Menu menu = new Menu("File");

save = new MenuItem("Save");

saveAs = new MenuItem("Save As");

load = new MenuItem("Load");

exit = new MenuItem("Exit");

menu.getItems().add(save);

menu.getItems().add(saveAs);

menu.getItems().add(load);

menu.getItems().add(exit);

this.getMenus().addAll(menu);

addActionHandlers();

}

private void addActionHandlers() {

save.setOnAction((ActionEvent event) -> {

try {

controller.saveToFile();

} catch (IOException ex) {

}

});

saveAs.setOnAction((ActionEvent event) -> {

try {

controller.saveAsToFile();

} catch (IOException ex) {

}

});

load.setOnAction((ActionEvent event) -> {

try {

controller.loadFromFile();

} catch (IOException ex) {

}

});

exit.setOnAction((ActionEvent event) -> {

controller.exitProgram();

});

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package view;

import javafx.animation.AnimationTimer;

import javafx.geometry.Insets;

import javafx.scene.image.ImageView;

import javafx.scene.layout.VBox;

public class SaveAnimationView {

private ClockTimer timer;

private ImageView bookImage;

private VBox imageBox;

private double y, w ;

private long old;

public SaveAnimationView(VBox imageBox) {

timer = new ClockTimer();

this.imageBox = imageBox;

initAnimation(imageBox);

}

private void initAnimation(VBox imageBox) {

bookImage = new ImageView(this.getClass().

getResource("/resources/library-books-t.png").

toString());

bookImage.setPreserveRatio(true);

bookImage.setVisible(false);

imageBox.getChildren().setAll(bookImage);

imageBox.setMouseTransparent(true);

}

public void startAnimation() {

y = 300.0;

w = 100.0;

old = 0;

imageBox.setPadding(new Insets(y, 10, 10, 350));

bookImage.setFitWidth(w);

bookImage.setVisible(true);

timer.start();

}

private class ClockTimer extends AnimationTimer{

@Override

public void handle(long now) {

// normal animation for super high fps

if (now > old + 10000000) {

old = now;

y -= 3;

w -= 1;

imageBox.setPadding(new Insets(y, 10, 10, 400));

bookImage.setFitWidth(w);

if (y < 40) {

bookImage.setVisible(false);

timer.stop();

}

}

}

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package view;

import controller.Controller;

import javafx.event.ActionEvent;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.control.Button;

import javafx.scene.control.ComboBox;

import javafx.scene.control.TextField;

import javafx.scene.layout.HBox;

public class SearchFieldView extends HBox {

private Controller controller;

private TextField searchField;

private ComboBox<String> searchComboBox;

private Button searchButton;

public SearchFieldView(Controller controller) {

super(30);

this.controller = controller;

initView();

}

private void initView() {

this.setAlignment(Pos.BASELINE\_LEFT);

this.setPadding(new Insets(5, 10, 10, 10));

searchField = new TextField();

searchField.setPromptText("Type here to search then press --->");

searchButton = new Button("Search");

searchComboBox = new ComboBox<>();

searchComboBox.getItems().addAll(

"Title",

"ISBN",

"Author"

);

searchComboBox.setPromptText("Title");

searchComboBox.setPrefWidth(90);

searchField.setPrefWidth(210);

searchButton.setPrefWidth(90);

addHandlers();

this.getChildren().addAll(searchComboBox, searchField, searchButton);

}

public String getComboText() {

return searchComboBox.getValue();

}

public String getSearched() {

return searchField.getText();

}

private void addHandlers() {

searchButton.setOnAction((ActionEvent event) -> {

controller.searchBook(getComboText(), getSearched());

});

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package view;

import controller.Controller;

import java.io.IOException;

import javafx.event.ActionEvent;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.control.Button;

import javafx.scene.layout.HBox;

public class TopHboxView extends HBox {

private Controller controller;

private Button addBook;

private Button removeBook;

private Button quickSave;

private Button refresh;

public TopHboxView(Controller controller) {

super(30);

this.controller = controller;

initView();

addEventHandlers();

}

private void initView() {

this.setAlignment(Pos.BASELINE\_LEFT);

this.setPadding(new Insets(10, 10, 5, 10));

addBook = new Button("Add Book");

removeBook = new Button("Remove Book");

quickSave = new Button("Save");

refresh = new Button("Refresh");

addBook.setMinWidth(90);

removeBook.setMinWidth(90);

quickSave.setMinWidth(90);

refresh.setMinWidth(90);

this.getChildren().addAll(addBook, removeBook, refresh, quickSave);

}

private void addEventHandlers() {

// med lambda expressions

addBook.setOnAction((ActionEvent event) -> {

controller.addBook();

});

removeBook.setOnAction((ActionEvent event) -> {

controller.removeBook();

});

refresh.setOnAction((ActionEvent event) -> {

controller.refresh();

});

quickSave.setOnAction((ActionEvent event) -> {

try {

controller.saveToFile();

} catch (IOException ex) {

}

});

}

}