Assignment 3: Film Database Application

Due date: Friday, 21st October 2016 at 12noon on BlackBoard.
This assignment is worth 10% of your final grade and has 100 marks in total.

Brief

Design and implement a film database application with a functional graphical user interface.

Program Interaction

Your application allows the user to display, add to and search a list of films:

- Read in film data from a text file (such as: title, director, rating, release year, cast list and genre)
- Save the database to a text file when exiting the application
- Add new films to the list via the GUI
- Delete existing films from the list via the GUI
- View a list of all films stored in the database
- Filter the list of films according to a criterion

Text file processing

Your application should handle text file access without crashing (via an uncaught exception). When the app starts it reads the film data from a text file. The text file can be whichever format you choose. The user's modification of the database through the GUI is saved back to the text file when the application exits.

Film List

Your application must display a list of the films in the database. Use appropriate Swing (**JComponent-based)** components to enable the user to input new films. The user may also remove films from the database.

Search Criteria

Your application must search across any combination of:

- Keyword search (e.g. name of title, cast member or director)
- Ratings (e.g. all films rated within a range)
- Release data (e.g. films released within a range of years)

Here are some examples of searches the user can perform through the GUI:

- All films directed by Clint Eastwood
- All films casting Kevin Bacon
- Action films from the 1980s
- All films casting Bruce Campbell between the dates 1980-1990 with more than a two stars rating.

Marking Scheme

Criteria:	Weight:	Grade A Grade Range: 100 ≥ x ≥ 80%	Grade B Grade Range: 80 > x ≥ 65%	Grade C Grade Range: 65 > x ≥ 50%	Grade D Grade Range: 50 > x ≥ 0%
Functionality Of the Database	**30%**	OOP paradigm consistently used for implementation of all database functionality	Inconsistent use of OOP paradigm but correct implementation of database functionality	Poor use of OOP paradigm	Absent database functionality or code does not compile
Functionality Of the view	**20%**	Excellent design of GUI with good choice of **JComponent** layout. View class designed with OOP paradigm	Inconsistent/poor design of GUI. View class design with OOP paradigm	Poor class design or flawed GUI design	Absent functionality for the view class or code does not compile
Functionality Of the controller	**20%**	Controller provides an elegant implementation of the database functionality via the GUI. All functionality is implemented	Controller provides basic implementation of the database functionality via the GUI. All functionality is implemented	Controller provides flawed implementation of the database functionality via the GUI. All functionality is implemented	Absent implementation of functionality or code does not compile
Code Quality: **-Whitespace** **-Naming** **-Reuse** **-Modularity** **-Encapsulation**	**20%**	Whitespace is comprehensively consistent. All naming is sensible and meaningful. Code reuse is present. Code is modular. Code is well encapsulated.	Whitespace is comprehensively consistent. Majority of naming is sensible. Code is modular. Code is encapsulated.	Whitespace is comprehensively consistent. Code has some modularity. Code has some encapsulation.	Whitespace is inconsistent and hence code is difficult to read.
Documentation Standards: **-Algorithms Commented** **-Javadoc**	**10%**	Entire codebase has comprehensive Javadoc commenting. Algorithms are well commented.	Majority of the codebase features Javadoc commenting. Majority of algorithms are commented.	Some Javadoc comments present. Some algorithms are commented.	No Javadoc comments present.

Javadoc Commenting

1. Your classes must have commenting of the form:

```
/**
  * Comment describing the class.
  * @author kjohnson studentnumber
 **/
```

2. All methods must be commented with appropriate Javadocs metatags. For example:

```
/**
 * A comment to describe the method
 * @param a parameter description
 * @return a description of the returned result
 * @author kjohnson studentnumber
 * */
```

Submission Instructions

1. Download and import the Java project **Assignment3** to your Eclipse workspace
2. Use the export feature in Eclipse to export your **Assignment3** resources to an archive file with the naming format: Lastname-firstName-studentID.zip.
3. Upload your archive file to the Blackboard system.

Late submissions will receive a grade of 0

An extension will only be considered with a Special Consideration Form approved by the School Registrar. These forms are available at the School of Engineering, Computer and Mathematical Science located in the WT Level 1 Foyer.

You will receive your marked assignment via the Blackboard Grade Centre. Please look over your entire assignment to make sure that it has been marked correctly. If you have any concerns, you must raise them with the lecturer. You have **one week** to raise any concerns regarding your mark. After that time, your mark cannot be changed.
Do not email the lecturer because you do not like your mark. Only go if you feel something has been marked incorrectly.

Authenticity

All work submitted must be unique and your own -
We use automated methods to detect academic integrity breaches.