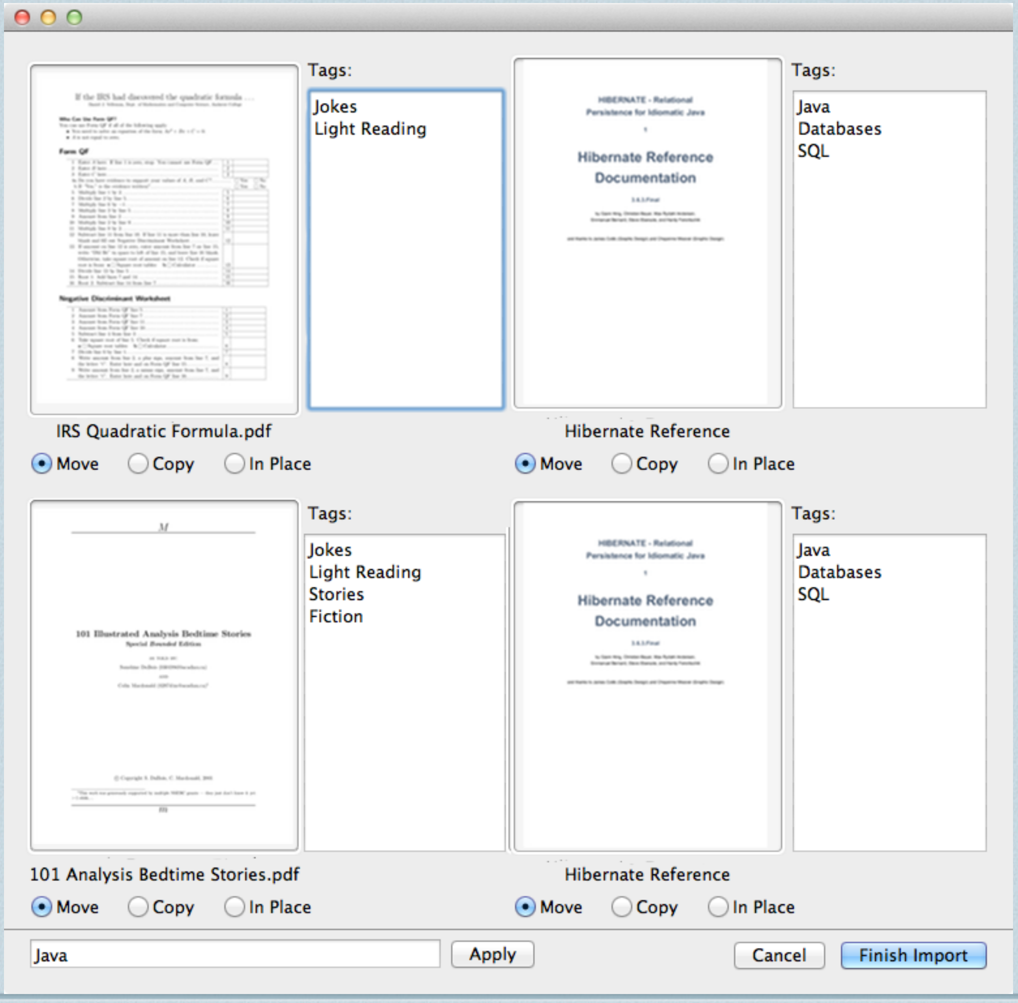
Biblio

- Dan Crankshaw
- Cain Lu
- Paul O'Neill
- Jiefeng Zhai

Motivation

- Strange File Names
- Music libraries, so why not text-based libraries?
- Non-hierarchical relationships
- Organize and view in one application

Importing



IRS Quadratic Formula.pdf

Tags:
Jokes
Light Reading

[✓ Move] Copy In Place

Hibernate Reference

Tags:
Java
Databases
SQL

[✓ Move] Copy In Place

101 Analysis Bedtime Stories.pdf

Tags:
Jokes
Light Reading
Stories
Fiction

[✓ Move] Copy In Place

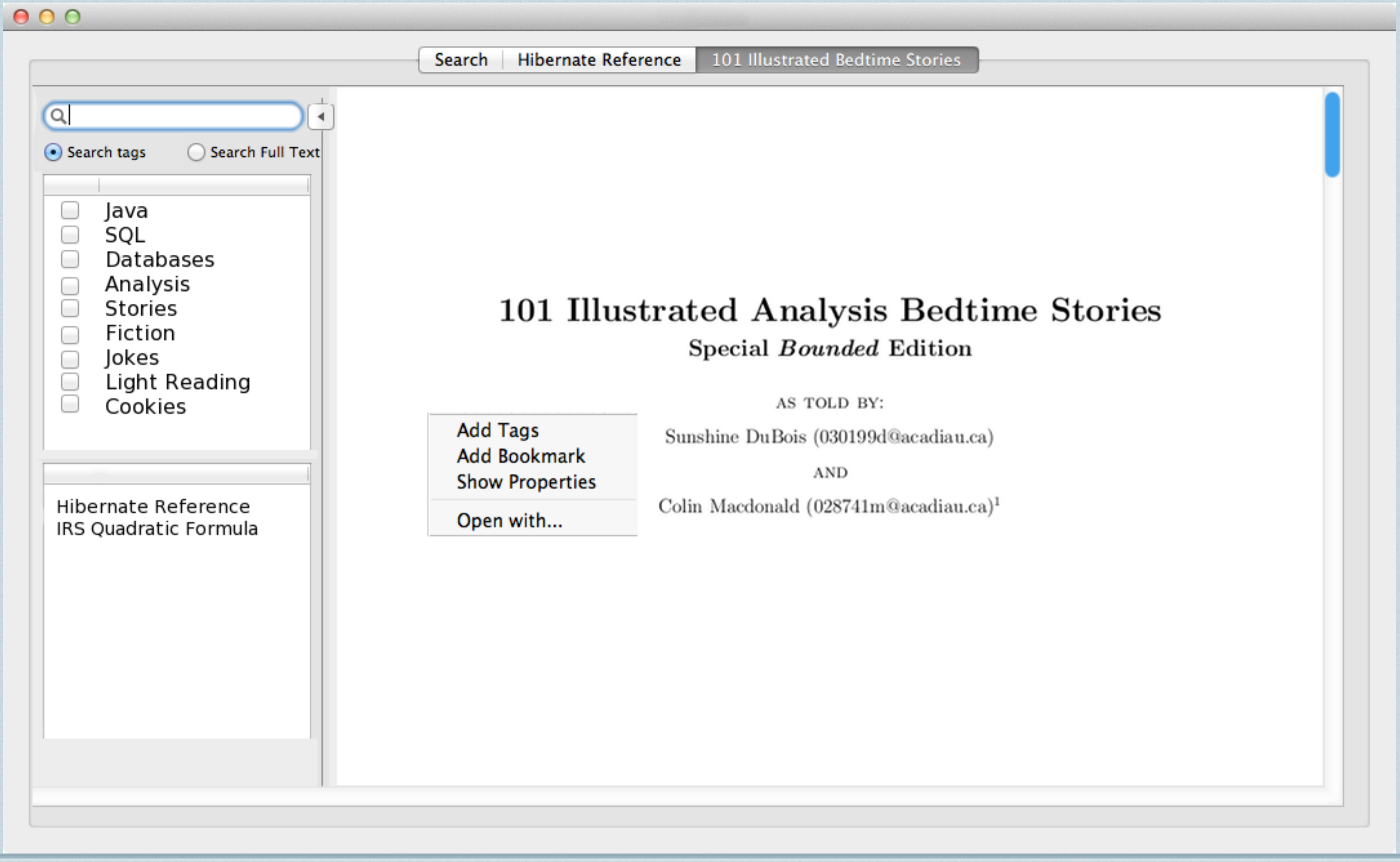
Hibernate Reference

Tags:
Java
Databases
SQL

[✓ Move] Copy In Place

Java

Reading



Hibernate Reference
IRS Quadratic Formula

101 Illustrated Analysis Bedtime Stories
Special *Bounded* Edition

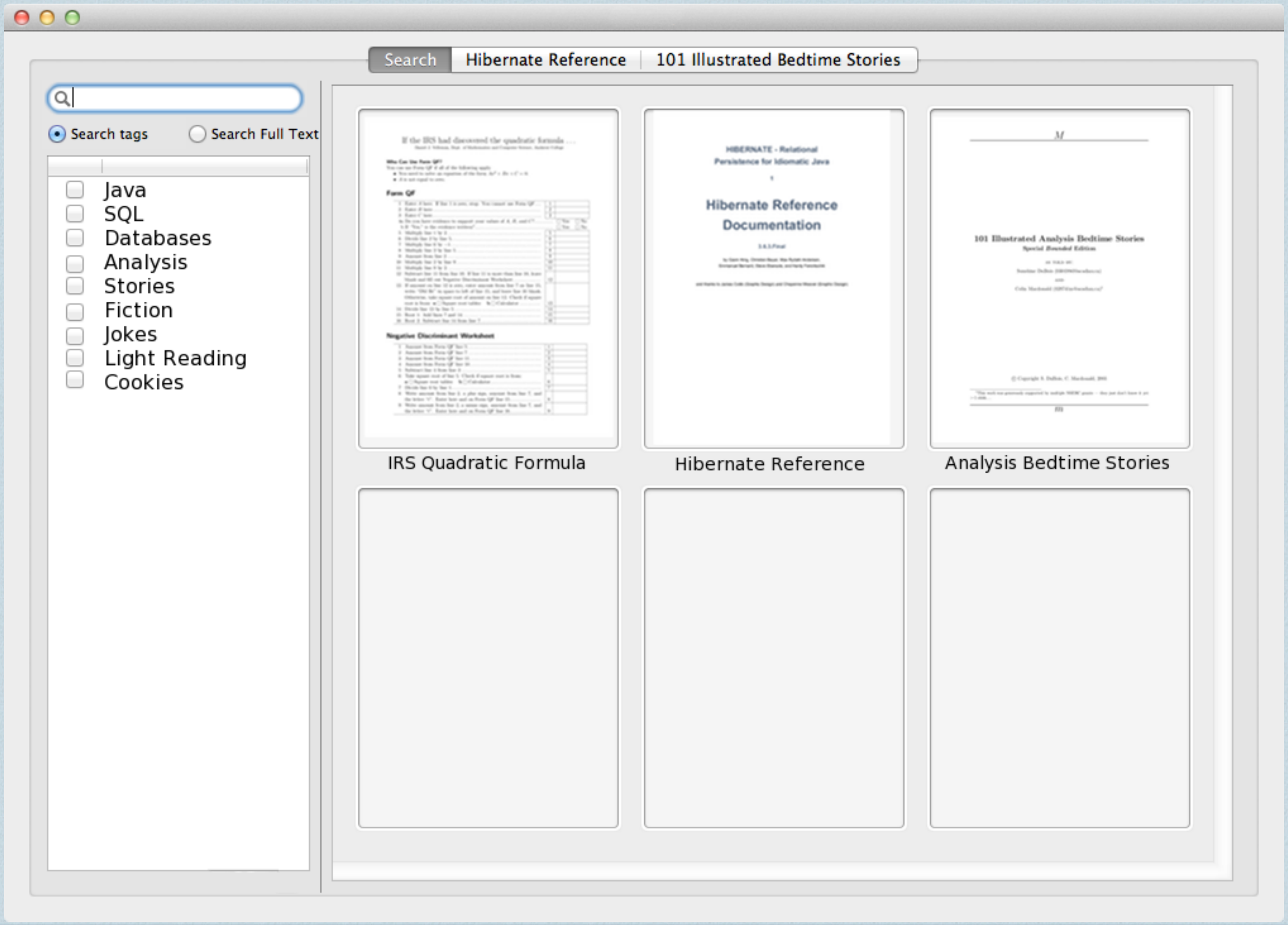
AS TOLD BY:

Sunshine DuBois (030199d@acadiau.ca)

AND

Colin Macdonald (028741m@acadiau.ca)[1]

Searching



Search | Hibernate Reference | 101 Illustrated Bedtime Stories

(● Search tags) ○ Search Full Text

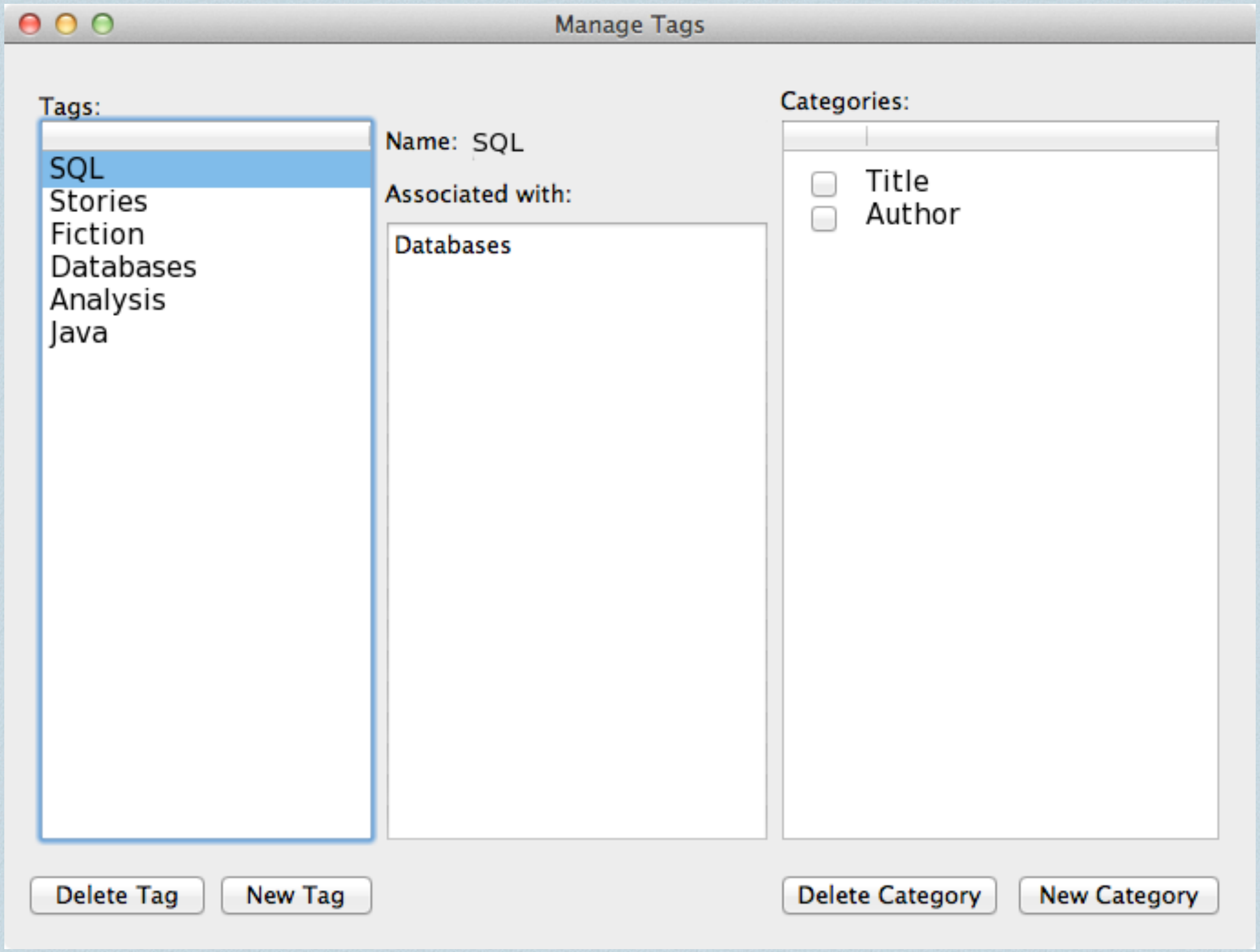
- [ ] Java
- [ ] SQL
- [ ] Databases
- [ ] Analysis
- [ ] Stories
- [ ] Fiction
- [ ] Jokes
- [ ] Light Reading
- [ ] Cookies

IRS Quadratic Formula

Hibernate Reference

Analysis Bedtime Stories

Tag Management



Manage Tags

Tags:

- SQL
- Stories
- Fiction
- Databases
- Analysis
- Java

Name: SQL

Associated with:

Databases

Categories:

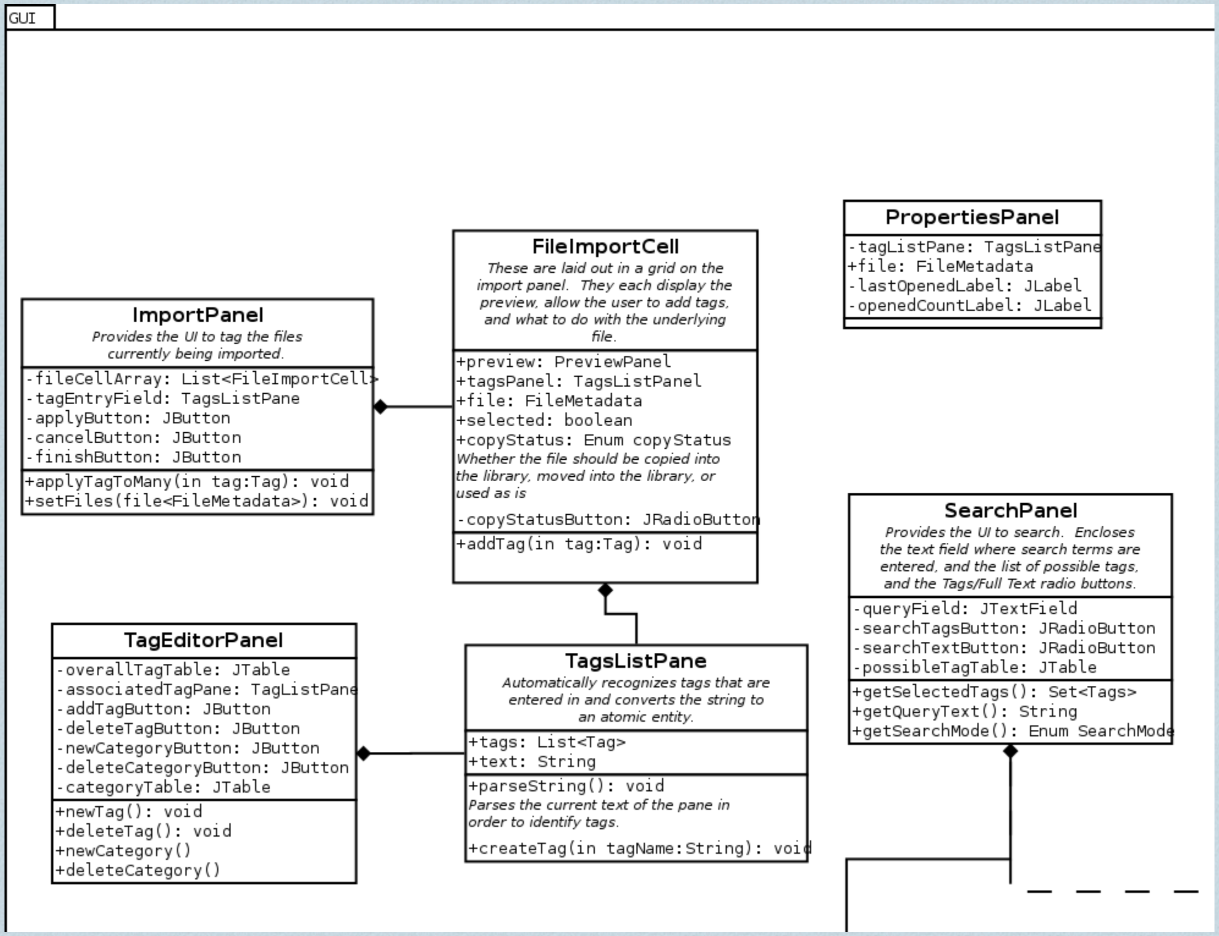
- [ ] Title
- [ ] Author

Hibernate Bug

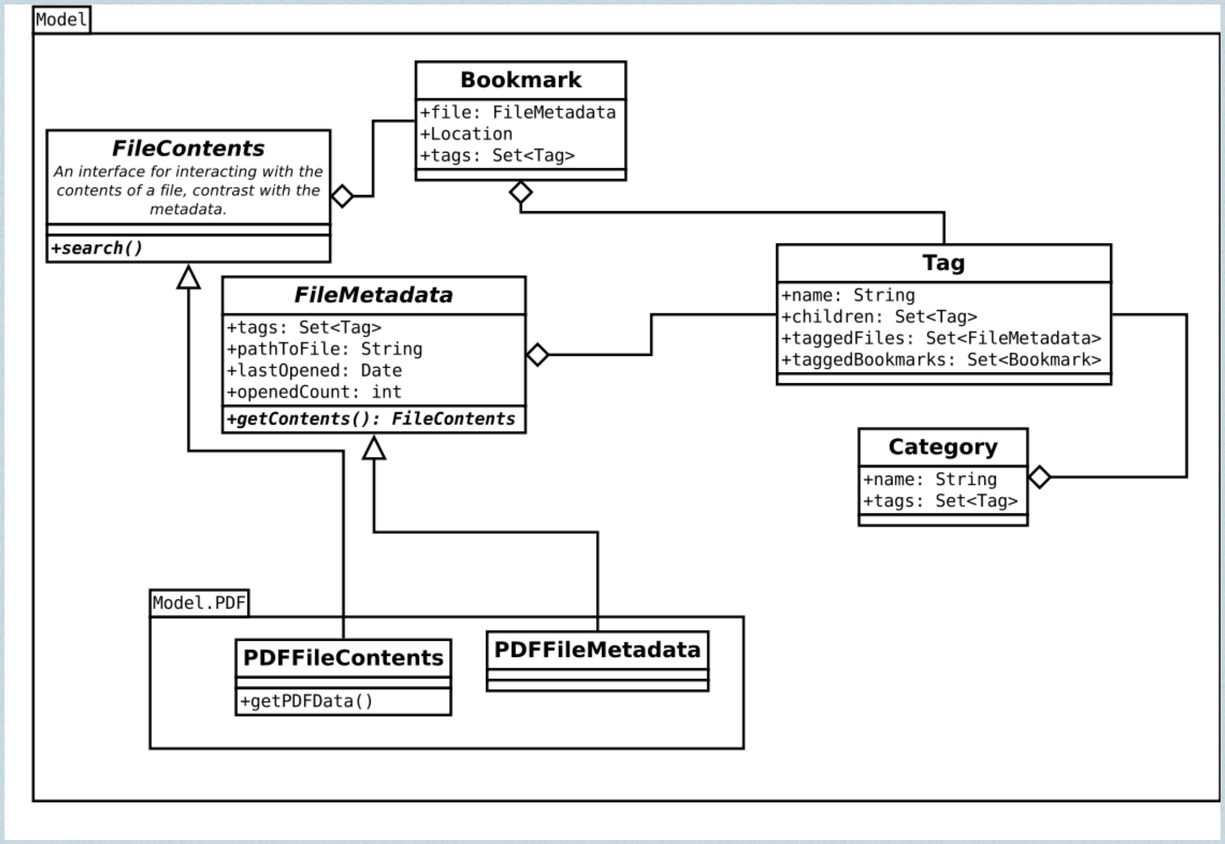
- When hibernate persists a set, it uses Object.hashCode() as the object identifier
- This can break contains() type methods on the returned set
- #HHH-3799

Demo!

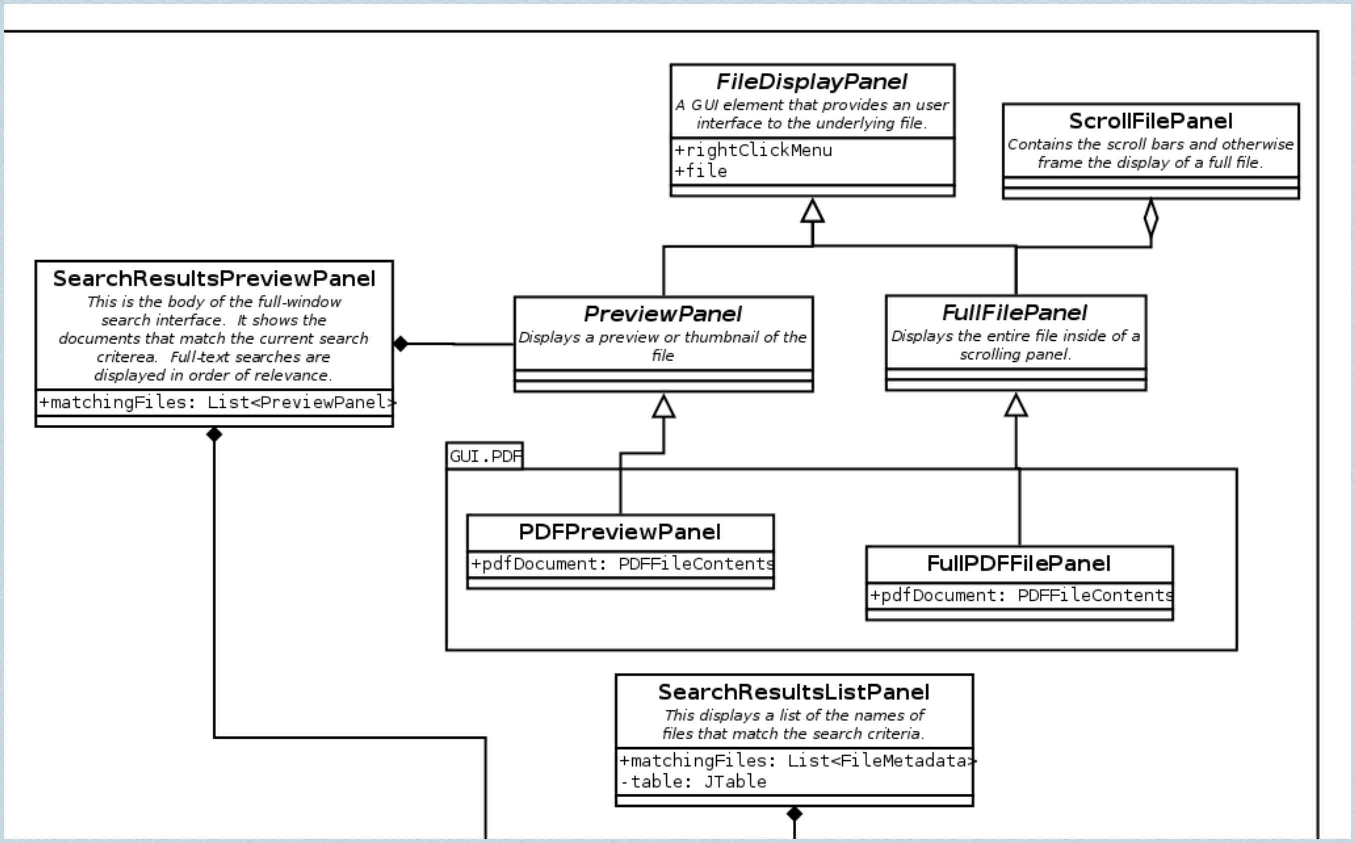
Code Overview: Import


GUI
ImportPanel
Provides the UI to tag the files currently being imported.
-fileCellArray: List<FileImportCell>
-tagEntryField: TagsListPane
-applyButton: JButton
-cancelButton: JButton
-finishButton: JButton
+applyTagToMany(in tag:Tag): void
+setFiles(file<FileMetadata>): void
FileImportCell
These are laid out in a grid on the import panel. They each display the preview, allow the user to add tags, and what to do with the underlying file.
+preview: PreviewPanel
+tagsPanel: TagsListPanel
+file: FileMetadata
+selected: boolean
+copyStatus: Enum copyStatus
Whether the file should be copied into the library, moved into the library, or used as is
-copyStatusButton: JRadioButton
+addTag(in tag:Tag): void
PropertiesPanel
-tagListPane: TagsListPane
+file: FileMetadata
-lastOpenedLabel: JLabel
-openedCountLabel: JLabel
TagEditorPanel
-overallTagTable: JTable
-associatedTagPane: TagListPane
-addTagButton: JButton
-deleteTagButton: JButton
-newCategoryButton: JButton
-deleteCategoryButton: JButton
-categoryTable: JTable
+newTag(): void
+deleteTag(): void
+newCategory()
+deleteCategory()
TagsListPane
Automatically recognizes tags that are entered in and converts the string to an atomic entity.
+tags: List<Tag>
+text: String
+parseString(): void
Parses the current text of the pane in order to identify tags.
+createTag(in tagName:String): void
SearchPanel
Provides the UI to search. Encloses the text field where search terms are entered, and the list of possible tags, and the Tags/Full Text radio buttons.
-queryField: JTextField
-searchTagsButton: JRadioButton
-searchTextButton: JRadioButton
-possibleTagTable: JTable
+getSelectedTags(): Set<Tags>
+getQueryText(): String
+getSearchMode(): Enum SearchMode

Code Overview: Model

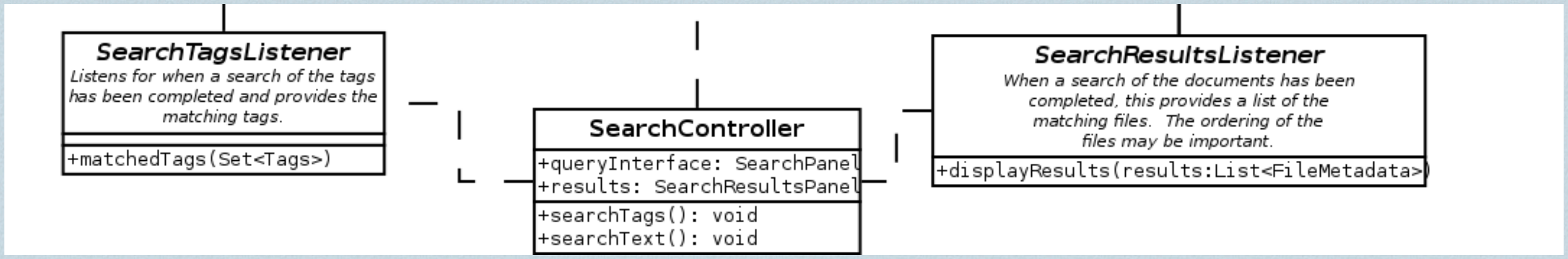


Code Overview: File Display



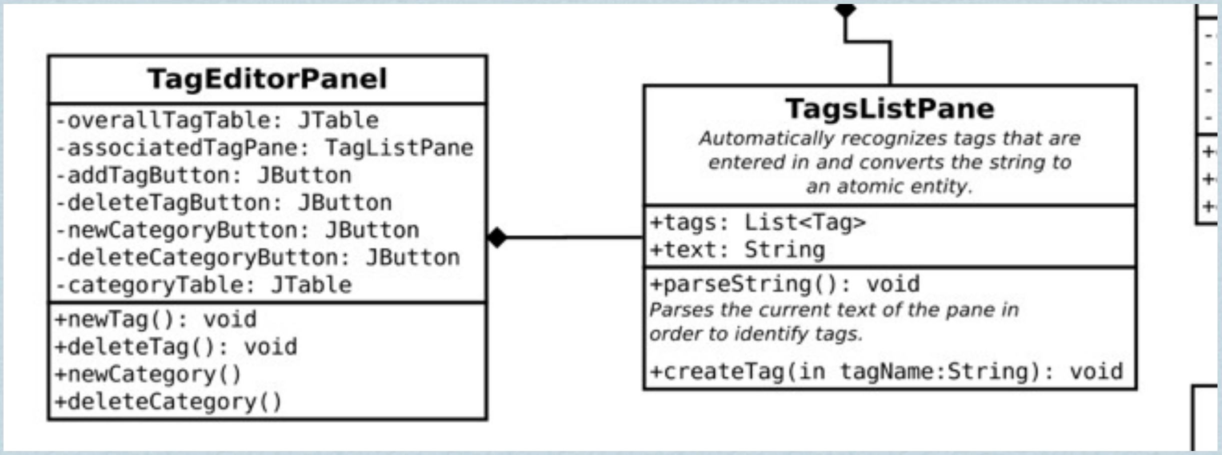
FileDisplayPanel
A GUI element that provides an user interface to the underlying file.
+rightClickMenu
+file
ScrollFilePanel
Contains the scroll bars and otherwise frame the display of a full file.
SearchResultsPreviewPanel
This is the body of the full-window search interface. It shows the documents that match the current search criterea. Full-text searches are displayed in order of relevance.
+matchingFiles: List<PreviewPanel>
PreviewPanel
Displays a preview or thumbnail of the file
FullFilePanel
Displays the entire file inside of a scrolling panel.
GUI.PDF
PDFPreviewPanel
+pdfDocument: PDFFileContents
FullPDFFilePanel
+pdfDocument: PDFFileContents
SearchResultsListPanel
This displays a list of the names of files that match the search criteria.
+matchingFiles: List<FileMetadata>
-table: JTable

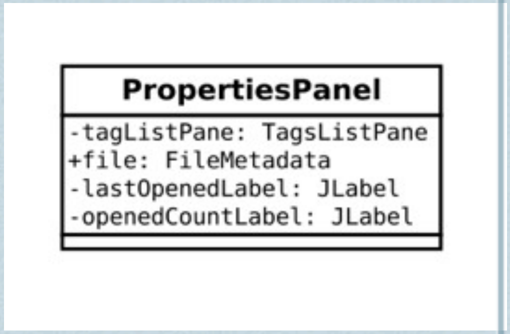
Code Overview: Search


SearchTagsListener
Listens for when a search of the tags has been completed and provides the matching tags.
+matchedTags(Set<Tags>)
SearchController
+queryInterface: SearchPanel
+results: SearchResultsPanel
+searchTags(): void
+searchText(): void
SearchResultsListener
When a search of the documents has been completed, this provides a list of the matching files. The ordering of the files may be important.
+displayResults(results:List<FileMetadata>)

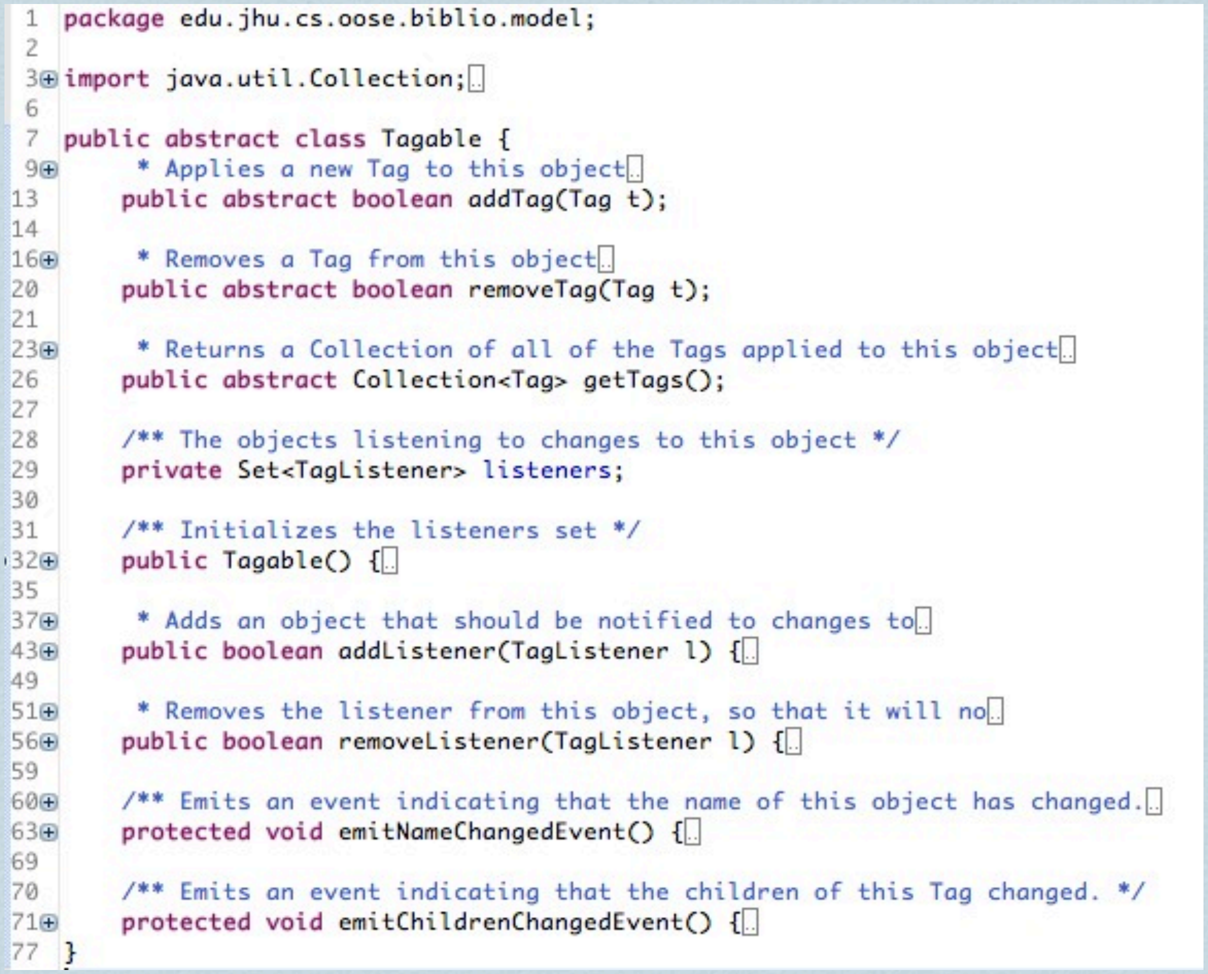
GUI

- Different parts of the GUI affect each other
- This is a Desktop application - updates needs to be instant
- We hand’t accounted for this in our initial design

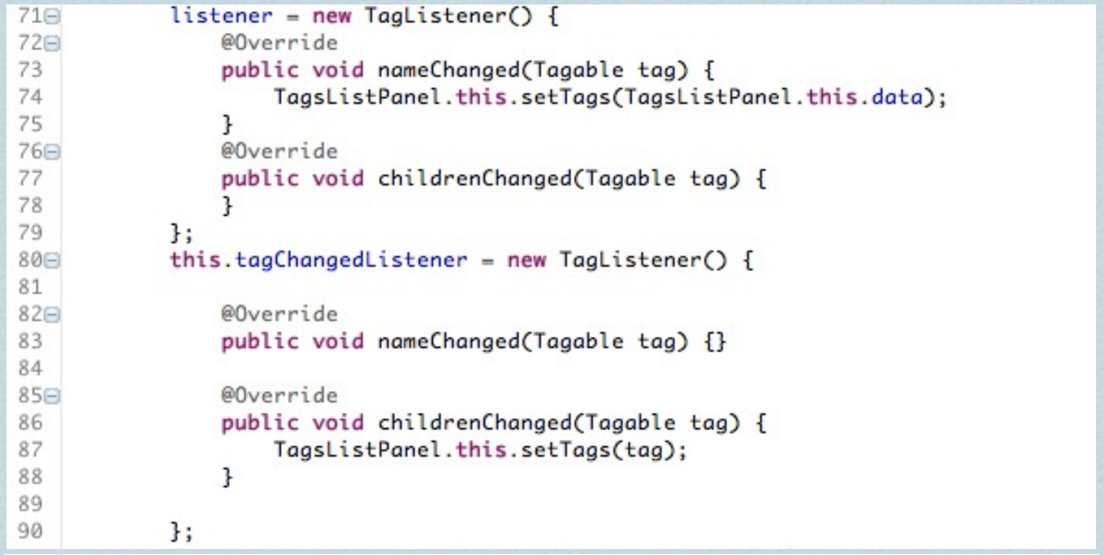

TagEditorPanel
-overallTagTable: JTable
-associatedTagPane: TagListPane
-addTagButton: JButton
-deleteTagButton: JButton
-newCategoryButton: JButton
-deleteCategoryButton: JButton
-categoryTable: JTable
+newTag(): void
+deleteTag(): void
+newCategory()
+deleteCategory()
TagsListPane
Automatically recognizes tags that are entered in and converts the string to an atomic entity.
+tags: List<Tag>
+text: String
+parseString(): void
Parses the current text of the pane in order to identify tags.
+createTag(in tagName:String): void



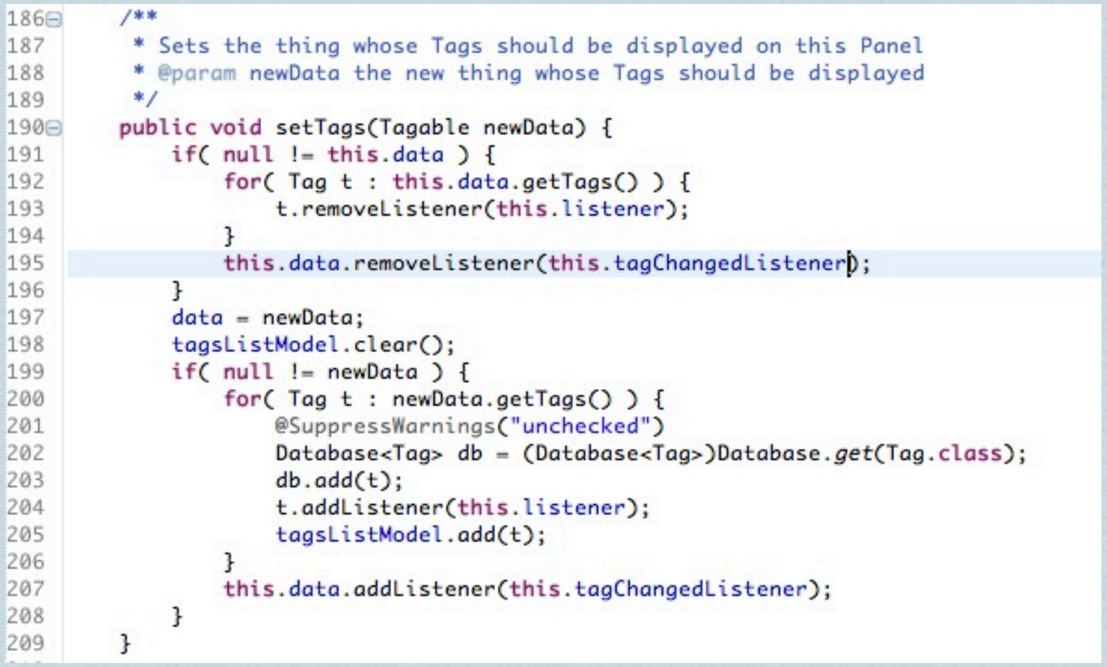
Solution!



```
package edu.jhu.cs.oose.biblio.model;

import java.util.Collection;

public abstract class Tagable {
     * Applies a new Tag to this object
    public abstract boolean addTag(Tag t);

     * Removes a Tag from this object
    public abstract boolean removeTag(Tag t);

     * Returns a Collection of all of the Tags applied to this object
    public abstract Collection<Tag> getTags();

    /** The objects listening to changes to this object */
    private Set<TagListener> listeners;

    /** Initializes the listeners set */
    public Tagable() {

     * Adds an object that should be notified to changes to
    public boolean addListener(TagListener l) {

     * Removes the listener from this object, so that it will no
    public boolean removeListener(TagListener l) {

    /** Emits an event indicating that the name of this object has changed.
    protected void emitNameChangedEvent() {

    /** Emits an event indicating that the children of this Tag changed. */
    protected void emitChildrenChangedEvent() {
}
```



```java
        listener = new TagListener() {
            @Override
            public void nameChanged(Tagable tag) {
                TagsListPanel.this.setTags(TagsListPanel.this.data);
            }
            @Override
            public void childrenChanged(Tagable tag) {
            }
        };
        this.tagChangedListener = new TagListener() {

            @Override
            public void nameChanged(Tagable tag) {}

            @Override
            public void childrenChanged(Tagable tag) {
                TagsListPanel.this.setTags(tag);
            }

        };
```

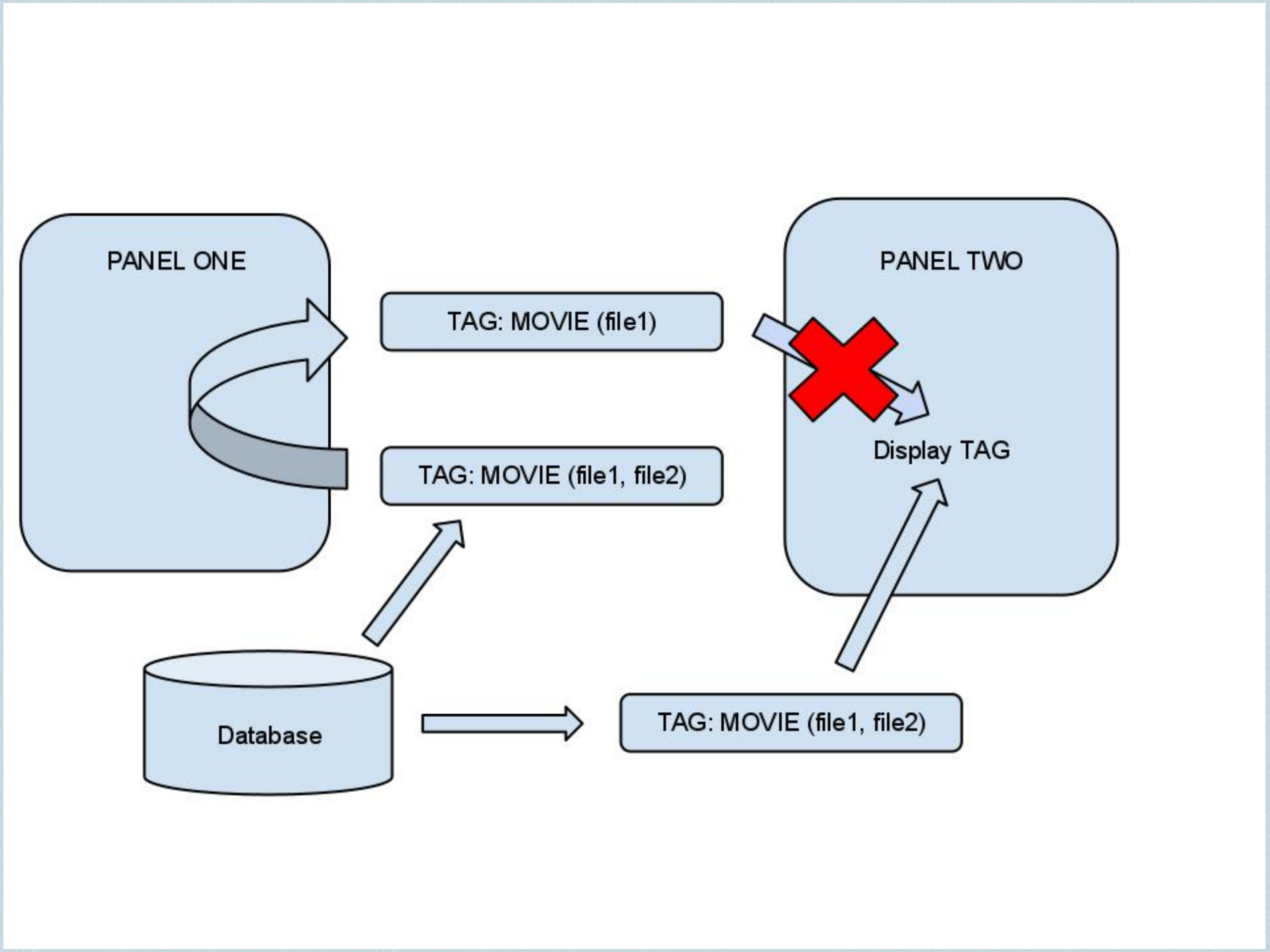


```java
    /**
     * Sets the thing whose Tags should be displayed on this Panel
     * @param newData the new thing whose Tags should be displayed
     */
    public void setTags(Tagable newData) {
        if( null != this.data ) {
            for( Tag t : this.data.getTags() ) {
                t.removeListener(this.listener);
            }
            this.data.removeListener(this.tagChangedListener);
        }
        data = newData;
        tagsListModel.clear();
        if( null != newData ) {
            for( Tag t : newData.getTags() ) {
                @SuppressWarnings("unchecked")
                Database<Tag> db = (Database<Tag>)Database.get(Tag.class);
                db.add(t);
                t.addListener(this.listener);
                tagsListModel.add(t);
            }
            this.data.addListener(this.tagChangedListener);
        }
    }
```

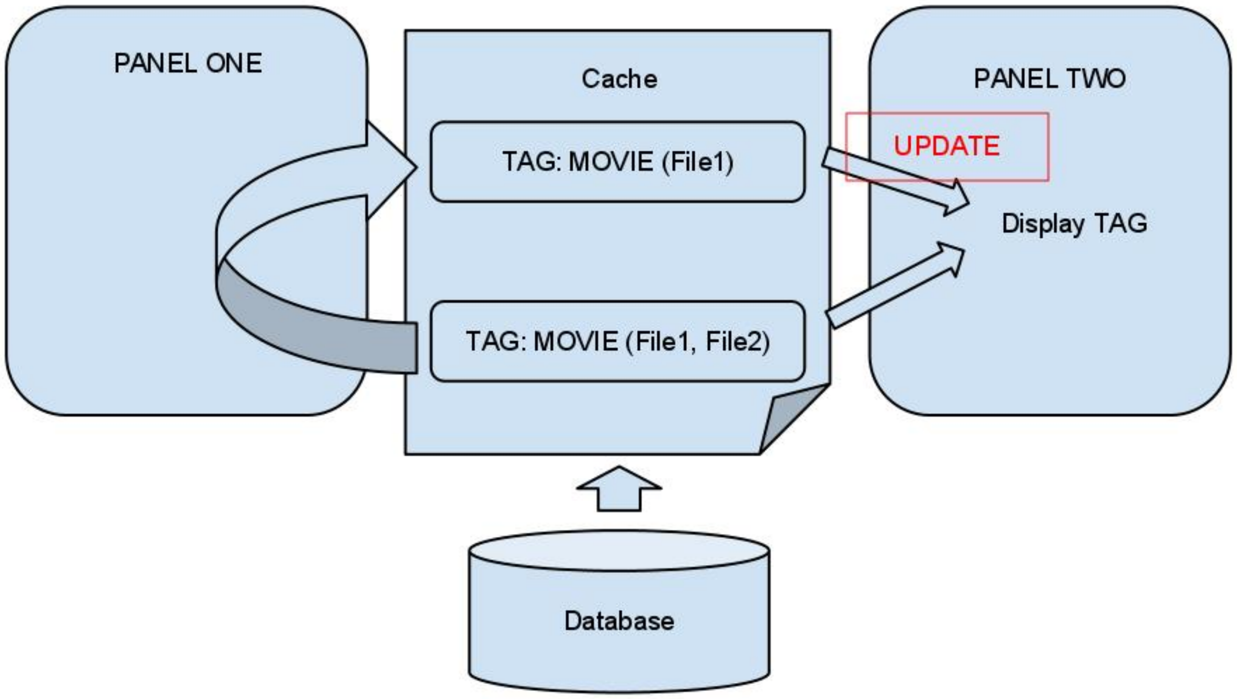
SearchManager

- Main interface between model and view
- Used to access DB – empty searches return everything
- Where 3rd party libraries get integrated into code
- Listeners on SearchManager have latest state

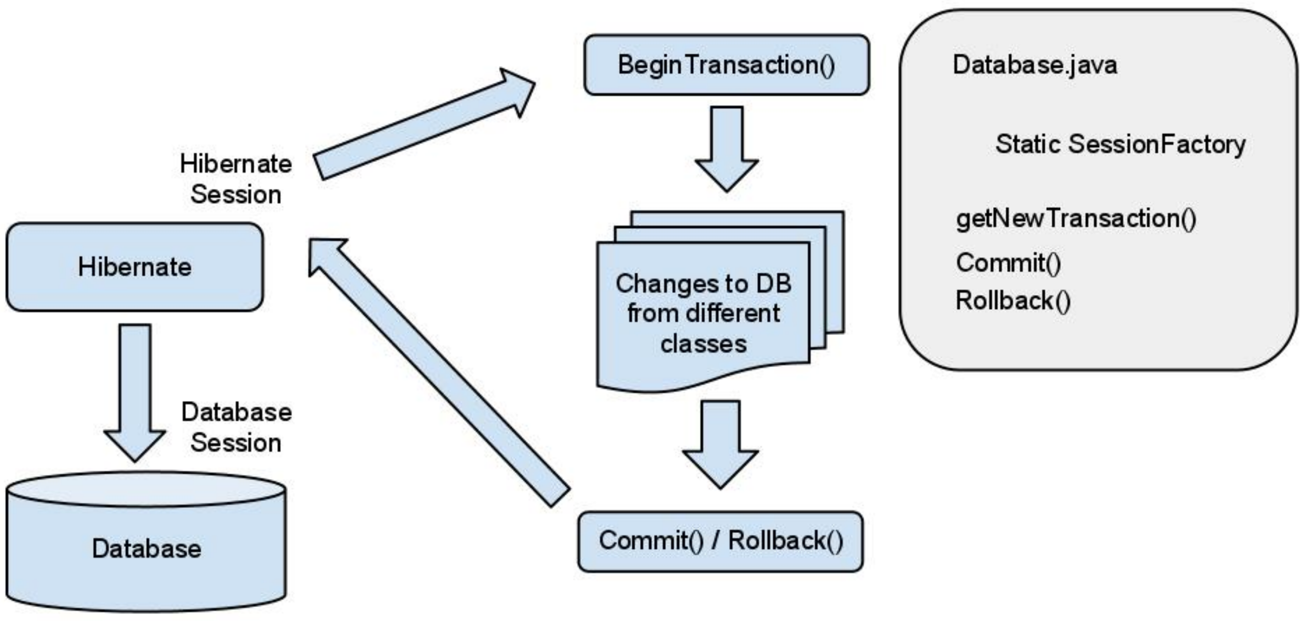
Database


PANEL ONE
TAG: MOVIE (file1)
TAG: MOVIE (file1, file2)
PANEL TWO
Display TAG
Database
TAG: MOVIE (file1, file2)

Database


PANEL ONE
Cache
TAG: MOVIE (File1)
TAG: MOVIE (File1, File2)
PANEL TWO
UPDATE
Display TAG
Database

Database

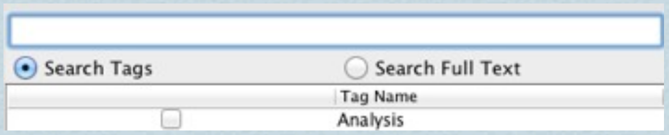

BeginTransaction()
Database.java
Static SessionFactory
getNewTransaction()
Commit()
Rollback()
Hibernate Session
Hibernate
Changes to DB from different classes
Database Session
Database
Commit() / Rollback()

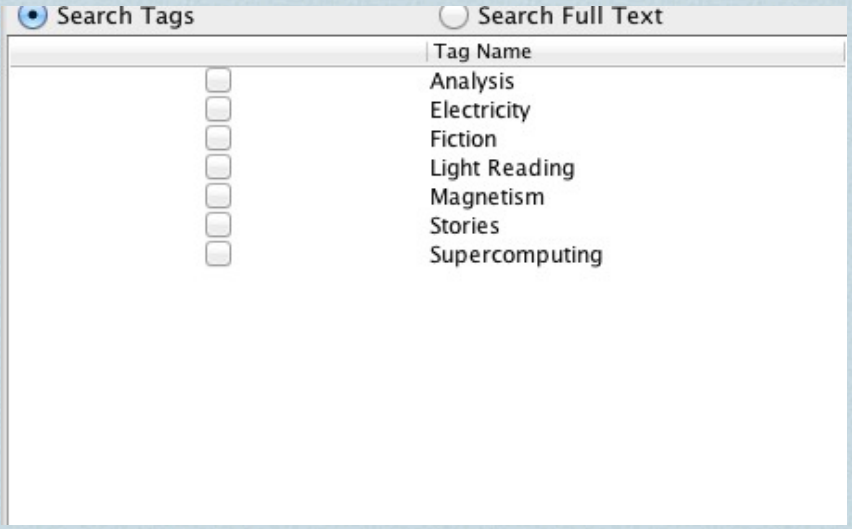
Searching

- We can search through:
  - The tags themselves
  - The text in files
  - For files tagged with certain Tags
  - For bookmarks tagged with Tags
- DRY!

Searching

- 2 Kinds of Searches





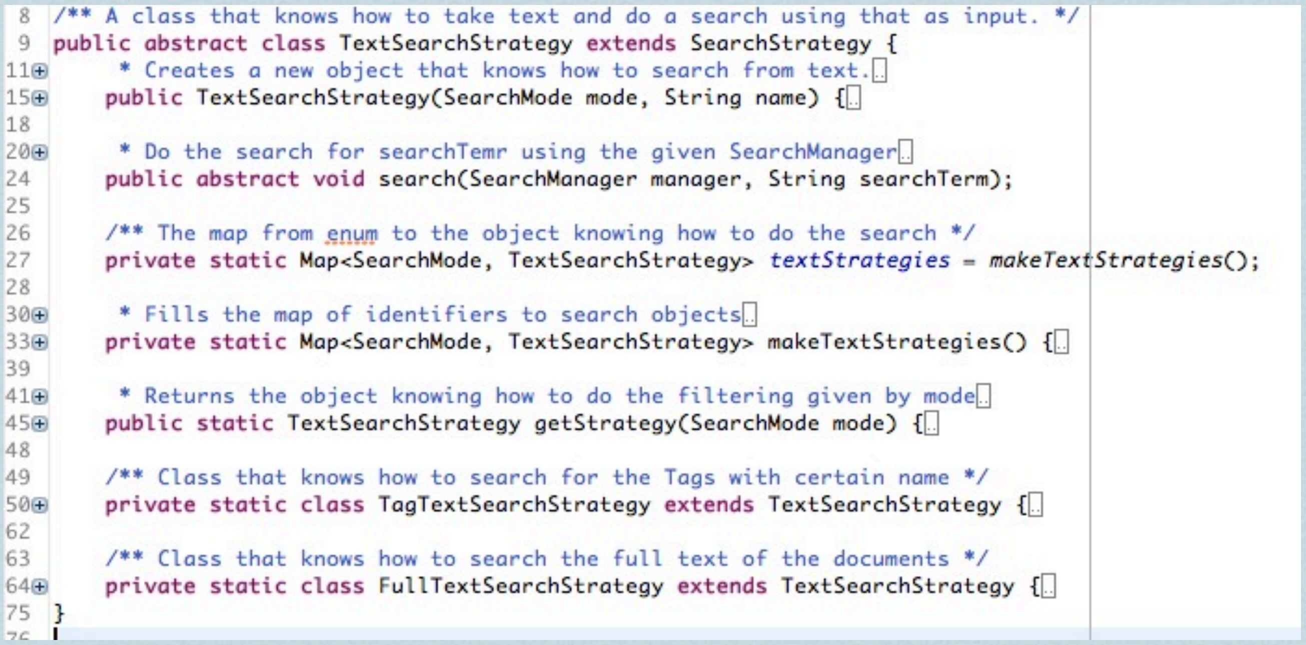
	Tag Name
☐	Analysis
☐	Electricity
☐	Fiction
☐	Light Reading
☐	Magnetism
☐	Stories
☐	Supercomputing

- Start from the Text Field

- Filtering using Tags

Solution!

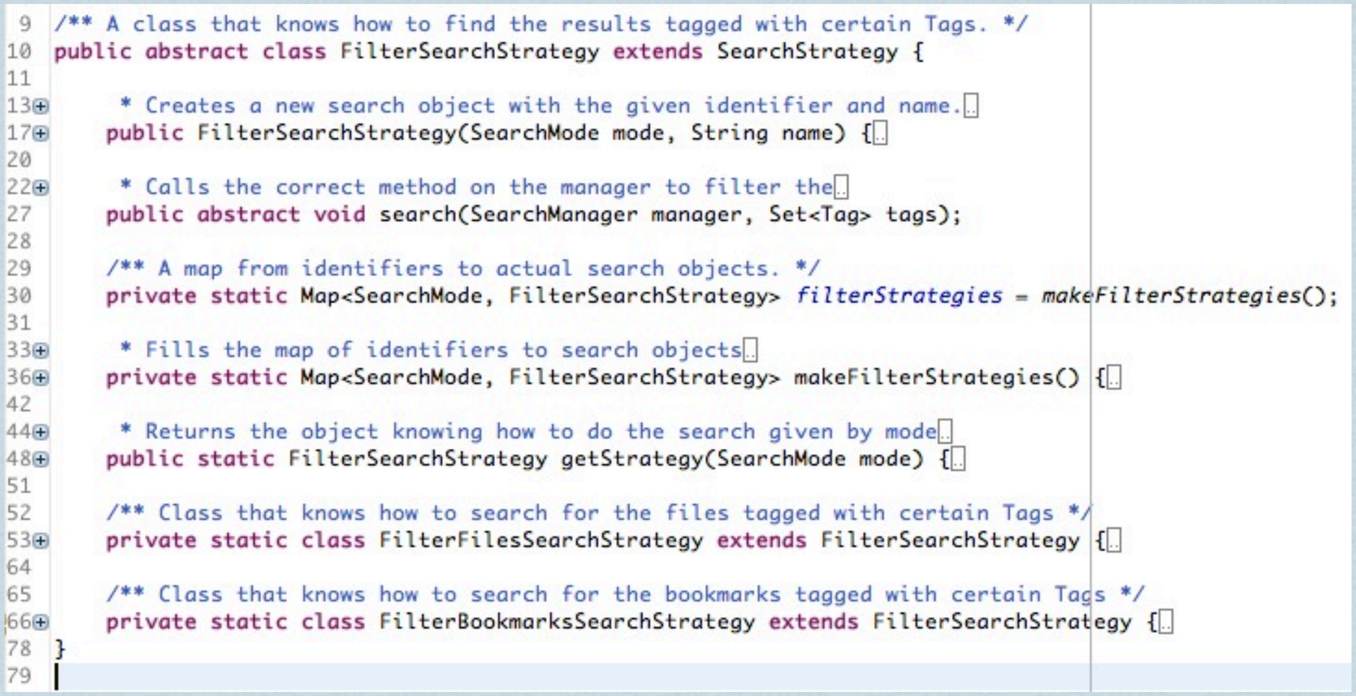
- Use the Strategy Pattern for each of them



```
/** A class that knows how to take text and do a search using that as input. */
public abstract class TextSearchStrategy extends SearchStrategy {
     * Creates a new object that knows how to search from text.
    public TextSearchStrategy(SearchMode mode, String name) {

     * Do the search for searchTemr using the given SearchManager
    public abstract void search(SearchManager manager, String searchTerm);

    /** The map from enum to the object knowing how to do the search */
    private static Map<SearchMode, TextSearchStrategy> textStrategies = makeTextStrategies();

     * Fills the map of identifiers to search objects
    private static Map<SearchMode, TextSearchStrategy> makeTextStrategies() {

     * Returns the object knowing how to do the filtering given by mode
    public static TextSearchStrategy getStrategy(SearchMode mode) {

    /** Class that knows how to search for the Tags with certain name */
    private static class TagTextSearchStrategy extends TextSearchStrategy {

    /** Class that knows how to search the full text of the documents */
    private static class FullTextSearchStrategy extends TextSearchStrategy {
}
```

Solution!

- Use the Strategy Pattern for each of them



```java
/** A class that knows how to find the results tagged with certain Tags. */
public abstract class FilterSearchStrategy extends SearchStrategy {

     * Creates a new search object with the given identifier and name.
    public FilterSearchStrategy(SearchMode mode, String name) {

     * Calls the correct method on the manager to filter the
    public abstract void search(SearchManager manager, Set<Tag> tags);

    /** A map from identifiers to actual search objects. */
    private static Map<SearchMode, FilterSearchStrategy> filterStrategies = makeFilterStrategies();

     * Fills the map of identifiers to search objects
    private static Map<SearchMode, FilterSearchStrategy> makeFilterStrategies() {

     * Returns the object knowing how to do the search given by mode
    public static FilterSearchStrategy getStrategy(SearchMode mode) {

    /** Class that knows how to search for the files tagged with certain Tags */
    private static class FilterFilesSearchStrategy extends FilterSearchStrategy {

    /** Class that knows how to search for the bookmarks tagged with certain Tags */
    private static class FilterBookmarksSearchStrategy extends FilterSearchStrategy {
}
```