Media Rental System

Software Design Document

Alexander R. Ramsey

August 12, 2017

SWEN 646

[1 INTRODUCTION 2](#_Toc490318295)

[1.1 Purpose 2](#_Toc490318296)

[1.2 Scope 2](#_Toc490318297)

[1.3 Overview 2](#_Toc490318298)

[1.4 Reference Material 2](#_Toc490318299)

[2 SYSTEM OVERVIEW 3](#_Toc490318300)

[3 SYSTEM ARCHITECTURE 4](#_Toc490318301)

[3.1 Architectural Design 4](#_Toc490318302)

[3.2 Decomposition Description 5](#_Toc490318303)

[3.3 Design Rationale 5](#_Toc490318304)

[4 DATA DESIGN 6](#_Toc490318305)

[4.1 Data Description 6](#_Toc490318306)

[4.2 Data Dictionary 6](#_Toc490318307)

[4.3 File Formatting 8](#_Toc490318308)

[5 COMPONENT DESIGN 9](#_Toc490318309)

[6 HUMAN INTERFACE DESIGN 23](#_Toc490318310)

[6.1 Overview of User Interface 23](#_Toc490318311)

[6.2 Screen Images 24](#_Toc490318312)

[6.3 Screen Objects and Actions 26](#_Toc490318313)

[7 REQUIREMENTS MATRIX 27](#_Toc490318314)

# 

# 

# 

# 1 INTRODUCTION

## 1.1 Purpose

This software design document describes the scope, definitions, architecture, and design for the creation of a simplistic Media Rental system. The document aims to extend the original Requirements Specification document, providing detailed descriptions of components and their inter relationships. The intended audience for this document is a software developer who is familiar with object oriented, aspect oriented, and UML design principles related to the Java programming language.

## 1.2 Scope

The Media Rental system will act as a basic cashier and tracking system for Customer rented Media. Supported Media includes Movie Dvd’s, AudioBook Cd’s, and Music Cd’s of varying Media Genres. When a Customer decides to Rent a Media, the Media Rental system will append the rental of the Media to the Customer Account File. The system will only track Media that is rented as it pertains to a customer. The system will also calculate rental fees for the Customer Account based upon a rate structure which varies per Media Type and Media Genre. When rental fees are calculated, an aspect class will respond by tallying a 10 cent fee per call of the method. The Aspect Class will return a Double with the total usage fee to be charged.

## 1.3 Overview

Section 1 of the document outlines the proceeding descriptor of a software application designed using Object Oriented, Aspect Oriented, and UML design methodologies. Section 2 of the document includes a System Overview which covers logical components and technologies utilized to produce the application. In Section 3, the Class Diagram provides a visual overview of the system data and its relationships and a descriptor of the relationships displayed. Section 4 addresses data storage through a file based system. Section 5 provides detailed descriptions and pseudo code of each method displayed in the Section 3 class diagram. Section 6 provides a User Interface design concept for data entry and menu navigation. Section 7 provides a requirements matrix which aims to address how each component in the design addresses the original requirements in the SRS document.

## 1.4 Reference Material

Software Requirements Specification and OOP Project Description for Media Rental System   
by University of Maryland University College for SWEN 646 Course

1.5 Definitions and Acronyms

* OOP: Object Oriented Programming. A design process for organizing blocks of information and instructions in programming as to achieve a higher order of extensibility and separation of concerns.
* AOP: Aspect Oriented Programming. An extension of the separation of concerns methodology deployable in the Java programming language where cross cutting concerns are addressed via a Joinpoint, a Pointcut, and through Weaving.
* Joinpoint: AOP term. An execution location in code where an advice method is triggered.
* Pointcut: AOP term. The method which identifies the join point in the code.
* Advice: AOP term. The method which contains the injected code at the join point.
* Weaving: AOP term. The action of injecting advice at join points using point cut and advice definitions into existing code base at the compiler level of operation.
* Media: A singular physical article which is capable of being rented.
* Customer Account: A billable and trackable User associated with an Account instance that retains Media.

# 2 SYSTEM OVERVIEW

The Media Rental System will be developed within the Java programming language. It will utilize Object Oriented and Aspect Oriented design principles to represent data as classes. Each class will have interdependent relationships. The object relationships will allow the system to organize data into logical groupings. Each class will have methods which will execute file system operations on User Account data. Each User Account data file will contain rented Media items of varying genre and fee rates. These files will use XML tags to store data. These tags will be parsable and readable when opened via the System. An Aspect class will act to record a tally of when fees are calculated via the calculateFee() method in each Genre/Media class.

The Media Rental System aims to act as a simple cashier record where customers will retain media for a 2 week period. Differing media types and genres determine the calculation of fees. Customer records are stored using files which contain Account User information and Media rental information. Media rentals are stored based upon inherited classes which differentiate the genres and media types which are offered. Below is a table which displays the varying types of Media available for rental:

|  |  |  |  |
| --- | --- | --- | --- |
| **Media Type** | **AudioBookCD** | **MusicCd** | **MovieDvd** |
| Sub Genre | Memoir | Country | Romance |
| SciFi | Classical | Drama |
| Romance |  | Documentary |

The software provider will be compensated for the usage of this application based upon every time a fee is calculated in each Media Type. The accumulated license fees will be tallied at 10 cents per operation of the calculateFee() method.

# 3 SYSTEM ARCHITECTURE

## 3.1 Architectural Design

A close up of text on a white background

Description generated with high confidenceBelow is a Class Diagram which represents the major components of the Media Rental System. The System utilizes Inheritance, Composition, and Aggregation to form relationships between the classes.

## 3.2 Decomposition Description

The Media Rental System presented in Section 3.1 is composed of classes which represent the Media Rental, User Account, User, and Media objects.

* The Media Rental class provides primary functionality to the end user interface. The MediaRental class has a 1 to 1 Composition parent relationship with the UserAccount class.
* The User Account class provides functionality for managing User Account details including the rental of none or many Media Objects and the storage of a single User object. The UserAccount class has a 1 to 1 Composition child relationship to the MediaRental class. The UserAccount class has a 1 to 1 Composition parent relationship to the User class. And, The UserAccount class has a 0 to Many Aggregation parent relationship to the Media class.
* The User class provides for basic customer information. The User class has a 1 to 1 Composition child relationship to the UserAccount class.
* The Media class provides for information pertaining to the rented media. The Media class has a 0 to Many Aggregation child relationship to the UserAccount class. The Media class has many child classes including Media delivery method classes, followed by their children, the Genre classes.
* The children of the Media class include Audio Book CD, Music CD, and Movie DVD classes. These classes differentiate Media based upon delivery method. Each delivery method has varying fee calculation.
* The children of the Media delivery method classes include Genres. These classes differentiate Media and its Delivery Method based upon the Genre of the media. Some Genres have differing fee calculation methods.
* The Aspect LicenseFee will target any time the calculateFee() method is called, incrementing a tally, which will be used to determine software license/usage fees.

## 3.3 Design Rationale

By utilizing an Object-Oriented approach to designing the Media Rental system, separation of concerns and extensibility can be maximized. The system was designed to be as simple as possible, fulfilling the requirements and scope of the original application proposal without over scoping. Despite the limited scope approach, by embracing an object-oriented design, the system is capable of being modified for future expansion and addition of functionality.

# 

# 4 DATA DESIGN

## 4.1 Data Description

Data in the system can be grouped into two categories, data which is pertinent to the system operation and data which is pertinent to the individual UserAccount class files. The method for data storage and retrieval will be file based. The UserAccount file will contain User Information, Media Information, and Fee Calculations.

## 4.2 Data Dictionary

| **Class** | **Field Name** | **Data Type** | **Data Format** | **Field Size** | **Description** |
| --- | --- | --- | --- | --- | --- |
| MediaRental | selectedAccount | Object | UserAccount |  |  |
| UserAccount | accountId | String |  | 20 | Current MS Time |
| UserAccount | user | Object | User |  |  |
| UserAccount | mediaRented | Array | Vector<Media> |  | Media Vector |
| User | firstName | String |  | 15 |  |
| User | lastName | String |  | 15 |  |
| User | email | String |  | 30 |  |
| Media | id | integer |  | 10 | Parameter |
| Media | title | String |  | 30 |  |
| Media | yearPublished | integer | YYYY | 4 |  |
| Media | rentalEndDate | String |  | 10 |  |
| Media | rentalFee | double | NN.DD | 10 |  |
| AudioBookCd | chapters | integer | NNNN | 4 |  |
| AudioBookCd | CURRENT\_YEAR\_FEE | double |  | 4 | 1.0 |
| AudioBookCd | CHAPTER\_FEE | double |  | 4 | 0.10 |
| MusicCd | fileSize | double |  | 4 |  |
| ­MusicCd | CURRENT\_YEAR\_FEE | double |  | 4 | 1.0 |
| MovieDvd | CURRENT\_YEAR\_FEE | double |  | 4 | 3.50 |
| MovieDvd | STANDARD\_FEE | double |  | 4 | 2.00 |
| CountryGenre | FILE\_SIZE\_FEE | double |  | 4 | 0.02 |
| ClassicalGenre | FILE\_SIZE\_FEE | double |  | 4 | 0.015 |
| RomanceDvdGenre | ADDITIONAL\_FEE | double |  | 4 | 0.25 |
| DramaGenre | ADDITIONAL\_FEE | double |  | 4 | 0.25 |
| LicenseFee <<Aspect>> | FEE | double |  |  | 0.10 |
| LicenseFee <<Aspect>> | tally | integer |  |  | Tally of calculateFee() calls |
| LicenseFee  <<Aspect>> | usageFee | double |  |  | Calculated usage fee by multiplying FEE by tally |

## 4.3 File Formatting

The MediaRental class contains methods to open, save, and delete UserAccount data in the form of a locally stored file format. The following section proposes a file format structure for storing applicable data by providing an example via the userAccount file.

Output: userAccount\_9999999999999.xml

<mediaRental>

<userAccount>

<accountId>9999999999999</accountId>

<user>

<firstName>Alex</firstName>

<lastName>Ramsey</lastName>

<email>alexramsey92@gmail.com</email>

</user>

<dramaGenre><id>1</id><title>Star Trek</title><yearPublished>2009</yearPublished><fileSize>10.00</fileSize><rentalEndDate>07/30/2017</rentalEndDate><rentalFee>2.00</rentalFee></dramaGenre>

</userAccount>

</mediaRental>

# 

# 5 COMPONENT DESIGN

**Class Name:** MediaRental

**Class Description:** The MediaRental class represents the highest level class for Media Rental operations. It acts as the initializer of UserAccount class details.

**Class Modifiers:** Public

**Class Inheritance:** None

**Class Attributes:**

private UserAccount selectedAccount

*This represents the loaded UserAccount file appended to a UserAccount object*

**Class Methods:**

public MediaRental()

Initializes UserAccount object.

public void addAccount(String firstName, String lastName, String email)

Creates new selectedAccount userAccount object

Calls selectedAccount.addUser(firstName, lastName, email)

public UserAccount openAccount(String id)

Calls selectedAccount.openAccount(id)

Returns selectedAccount object

public void saveAccount(UserAccount selectedAccount)

Sets account file name to match format: “userAccount\_accountId.xml”

Begins printwriter stream

Outputs mediaRental, userAccount, user tags

Outputs user information from selectedAccount object

Outputs closing user tag

If selectedAccount media is null

Output null

Else

For media objects in media

Set mediaString equal to media.convertToString()

Output mediaString

Outputs closing userAccount and mediaRental tags

Handles file operation exceptions

public deleteAccount()

If selectedAccount is null

Output “No account available to delete”

Else

Set filename String equal to selectedAccount accountId format

Remove selectedAccount file from file system

Sets selectedAccount object to null

Handles file delete exception

public getSelectedAccount()

Returns selectedAccount object.

public setSelectedAccount(UserAccount : userAccount)

Sets selectedAccount object equal to parameter UserAccount’

public double licenseFeeTotal()

Returns license fee value, default at 0.00.

**Class Name:** UserAccount

**Class Description:** The UserAccount class handles the UserAccount file and its contained data including the User data and Media array.

**Class Modifiers:** Public

**Class Inheritance:** None

**Class Attributes:**

private String accountId

*Auto Increment value for retrieval and data tracking purposes*

private User user

*1 to 1 User object contains the Customer User details*

private Vector<Media> media

*One to many Vector containing Media objects*

**Class Methods:**

public UserAccount()

Set accountId equal to current time in milliseconds

Initialize new User object

Initialize new Media vector

public void openAccount(String id)

If parameter accountId equals null

Print console message stating account id required

Else

Create file input stream pointed to xml file with parameter accountId value

Create buffered reader stream

While Lines of file exist

If line contains accountId

Set accountId equal to accountId from current line

Call setAccountId(accoundId)

Else if line contains User

Call User(line)

Else if line contains MemoirGenre

Call MemoirGenre(line)

Call rentMedia() on current media type

Else if line contains SciFiGenre

Call SciFiGenre(line)

Call rentMedia() on current media type

Else if line contains RomanceCdGenre

Call RomanceCdGenre(line)

Call rentMedia() on current media type

Else if line contains CountryGenre

Call CountryGenre(line)

Call rentMedia() on current media type

Else if line contains ClassicalGenre

Call ClassicalGenre(line)

Call rentMedia() on current media type

Else if line contains RomanceDvdGenre

Call RomanceDvdGenre(line)

Call rentMedia() on current media type

Else if line contains DramaGenre

Call DramaGenre(line)

Call rentMedia() on current media type

Else if line contains DocumentaryGenre

Call DocumentaryGenre(line)

Call rentMedia() on current media type

Close buffered reader and file reader stream

Handle File reader exceptions

public void addUser(String firstName, String lastName, String email)

Set User equal to new User(firstName, lastName, email)

public void rentMedia(Media media)

Adds new Media data to selectedAccount mediaRented Media Vector

public void returnMedia(Integer id)

For media in selectedAccount

If this.media id equals parameter id

Remove media from media array

public void renewMedia(Integer id)

Find media in selectedAccount.getMediaRented with matching parameter id

Sets rentalEndDate for parameter media id in selectedAccount media vector by adding 14 days to existing value

Calculates new rentalFee  
 Executes saveAccount() method

public String getAccountId()

Returns accountId String

public void setAccountId(String accountId)

Sets accountId String for passed parameter

public User getUser()

Returns User object

public Vector<Media> getMedia()

Returns Vector of Media data

**Class Name:** User

**Class Description:** The User class handles the basic UserAccount customer information. It is associated with a 1 to 1 aggregate relationship to each UserAccount.

**Class Modifiers:** Public

**Class Inheritance:** None

**Class Attributes:**

private String firstName

private String lastName

private String email

**Class Methods:**

public User(firstName: String, lastName : String, email : String)

Validates input values

Initializes User object with input values

Handles invalid input exception

public User(String line)

Sets line substring of firstName equal to firstName  
Sets line substring of lastName equal to lastName  
Sets line substring of email equal to email

public User()

Initializes User object values to null

public getFirstName()

Returns firstName value

public setFirstName(firstName : String)

Sets firstName value

public getLastName()

Returns lastName value

public setLastName(lastName : String)

Sets lastName value

public getEmail()

Returns email value

public setEmail(email : String)

Sets email value

**Class Name:** Media

**Class Description:** The Media class handles the individual Media objects which are stored in the UserAccount Media array.

**Class Modifiers:** Public Abstract

**Class Inheritance:** The Media class can be inherited by its child classes, AudioBookCd, MusicCd, and MovieDvd.

**Class Attributes:**

protected Integer id  
*Auto increment value for data retrieval and tracking*

protected String title  
*Title of Media*

protected Integer yearPublished  
*Year Media was published*

protected String rentalEndDate   
*When rented Media is to be returned*

protected Double rentalFee  
*Calculated fee of rented Media*

**Class Methods:**

public Media()

Initializes Media object values to null

public Media(Integer id, String title, Integer yearPublished)

Sets rentalEndDate equal to 14 days from current date time

Initializes Media object with input values

public abstract String convertToString()

Abstract method for child objects.

public Media(String line)

Method for child objects.

public Integer getId()

Returns Id value

public void setId(Integer id)

Sets Id value

public getTitle()

Returns title value

public setTitle(title : String)

Sets title value

public getYearPublished()

Returns yearPublished value

public setYearPublished(year : integer)

Sets yearPublished value

public getFileSize()

Returns fileSize value

public setFileSize(size : double)

Sets fileSize value

public getRentalEndDate()

Returns rentalEndDate value

public setRentalEndDate(date : Calendar)

Sets rentalEndDate value

public void abstract calculateFee()  
Override by child classes to calculate fees of Media rentals

**Class Name:** AudioBookCd

**Class Description:** The AudioBookCd class differentiates Media by adding chapters and by modifying the calculation of fees.

**Class Modifiers:** Public

**Class Inheritance:** Inherits Media Class methods and attributes.

**Class Attributes:**

protected Integer chapters

*Media type specific value for calculating fees*

public static final double CURRENT\_YEAR\_FEE = 1.00

public static final double CHAPTER\_FEE = 0.10

**Class Methods:**

public AudioBookCd(Integer id, String title, Integer yearPublished, Integer chapters)

Calls Parent constructor

Sets chapters to chapters parameter  
 Calls calculateFee()

public AudioBookCd(String line)

Calls Parent constructor

Sets value of id equal to substring between <id> tag

Sets value of title equal to substring between <title> tag  
 Sets value of yearPublished to substring between <yearPublished> tag  
 Sets value of rentalFee to substring between <rentalFee> tag

Sets value of rentalEndDate to substring between <rentalEndDate> tag

Sets value of chapters to substring between <chapters> tag

public String convertToString()

Sets className String equal to Simple name of class

Sets mediaString value to concatenation of

className value as an opening xml tag

id xml tags wrapping id value

title xml tags wrapping title value

yearPublished xml tags wrapping yearPublished value

chapters xml tags wrapping chapters value

rentalEndDate xml tags wrapping rentalEndDate value

rentalFee xml tags wrapping rentalFee value

className value as a closing xml tag

Returns mediaString

public Integer getChapters()

Returns value of chapters

public void setChapters(Integer chapters)

Sets value of chapters

public void calculateFee()

If yearPublished parameter equals Calendar Year

Set rentalFee equal to chapters parameter times CHAPTER\_FEE and Add CURRENT\_YEAR FEE

Else

Set rentalFee equal to chapters parameter times CHAPTER\_FEE

**Class Name:** MusicCd

**Class Description:** The MusicCd class differentiates Media by adding length and by modifying the calculation of fees.

**Class Modifiers:** Public

**Class Inheritance:** Inherits Media Class methods and attributes.

**Class Attributes:**

protected Double fileSize

*Media type specific value for calculating fees*

public static final double CURRENT\_YEAR\_FEE = 1.00

**Class Methods:**

public MusicCd(Integer id, String title, Integer yearPublished, Double fileSize)

Calls Parent constructor  
 Calls calculateFee()

public MusicCd(String line)

Calls Parent constructor

Sets value of id equal to substring between <id> tag

Sets value of title equal to substring between <title> tag  
 Sets value of yearPublished to substring between <yearPublished> tag

Sets value of fileSize to substring between <fileSize> tag  
 Sets value of rentalFee to substring between <rentalFee> tag

Sets value of rentalEndDate to substring between <rentalEndDate> tag

public String convertToString()

Sets className String equal to Simple name of class

Sets mediaString value to concatenation of

className value as an opening xml tag

id xml tags wrapping id value

title xml tags wrapping title value

yearPublished xml tags wrapping yearPublished value

fileSize xml tags wrapping fileSize value

rentalEndDate xml tags wrapping rentalEndDate value

rentalFee xml tags wrapping rentalFee value

className value as a closing xml tag

Returns mediaString

public void calculateFee()  
 Does not return any calculated value.

**Class Name:** MovieDvd

**Class Description:** The MovieDvd class differentiates Media by modifying the calculation of fees.

**Class Modifiers:** Public

**Class Inheritance:** Inherits Media Class methods and attributes.

**Class Attributes:**

public static final CURRENT\_YEAR\_FEE = 3.5

public static final STANDARD\_FEE = 2.0

**Class Methods:**

public MovieDvd(Integer id, String title, Integer yearPublished)

Calls Parent constructor  
 Calls calculateFee()

public MovieDvd(String line)

Calls Parent constructor

Sets value of id equal to substring between <id> tag

Sets value of title equal to substring between <title> tag  
 Sets value of yearPublished to substring between <yearPublished> tag  
 Sets value of rentalFee to substring between <rentalFee> tag

Sets value of rentalEndDate to substring between <rentalEndDate> tag

public String convertToString(){

Sets className String equal to Simple name of class

Sets mediaString value to concatenation of

className value as an opening xml tag

id xml tags wrapping id value

title xml tags wrapping title value

yearPublished xml tags wrapping yearPublished value

rentalEndDate xml tags wrapping rentalEndDate value

rentalFee xml tags wrapping rentalFee value

className value as a closing xml tag

Returns mediaString

public void calculateFee()

If yearPublished parameter equals Calendar Year

Set rentalFee equal to CURRENT\_YEAR\_FEE

Else

Set rentalFee equal to STANDARD\_FEE

**Class Name:** MemoirGenre

**Class Description:** The MemoirGenre class provides a descriptor for specific AudioBookCd media types

**Class Modifiers:** Public

**Class Inheritance:** Inherits AudioBookCd Class methods and attributes

**Class Attributes:** None

**Class Methods:**

public MemoirGenre(Integer id, String title, Integer yearPublished, Integer chapters)

Call Parent Constructor with Parameters

public MemoirGenre(line)

Call Parent Constructor with Parameters

**Class Name:** SciFiGenre

**Class Description:** The SciFiGenre class provides a descriptor for specific AudioBookCd media types

**Class Modifiers:** Public

**Class Inheritance:** Inherits AudioBookCd Class methods and attributes

**Class Attributes:** None

**Class Methods:**

public SciFiGenre(Integer id, String title, Integer yearPublished, Integer chapters)

Call Parent Constructor with Parameters

public SciFiGenre(line)

Call Parent Constructor with Parameters

**Class Name:** RomanceCdGenre

**Class Description:** The RomanceCdGenre class provides a descriptor for specific AudioBookCd media types

**Class Modifiers:** Public

**Class Inheritance:** Inherits AudioBookCd Class methods and attributes

**Class Attributes:** None

**Class Methods:**

public RomanceCdGenre(Integer id, String title, Integer yearPublished, Integer chapters)

Call Parent Constructor with Parameters

public RomanceCdGenre(line)

Call Parent Constructor with Parameters

**Class Name:** CountryGenre

**Class Description:** The CountryGenre class provides a descriptor for specific MusicCd media types and Calculates Fees based on Genre specific requirements

**Class Modifiers:** Public

**Class Inheritance:** Inherits MusicCd Class methods and attributes

**Class Attributes:**

public static final double FILE\_SIZE\_FEE = 0.02

**Class Methods:**

public CountryGenre(Integer id, String title, Integer yearPublished, Double fileSize)

Call Parent Constructor with Parameters

public CountryGenre(String line)

Call Parent Constructor with Parameters

Call calculateFee()

public void calculateFee()

If yearPublished parameter equals Calendar Year

Set rentalFee equal to fileSize times FILE\_SIZE\_FEE plus CURRENT\_YEAR\_FEE

Else

Set rentalFee equal to fileSize times FILE\_SIZE\_FEE

**Class Name:** ClassicalGenre

**Class Description:** The ClassicalGenre class provides a descriptor for specific MusicCd media types and Calculates Fees based on Genre specific requirements

**Class Modifiers:** Public

**Class Inheritance:** Inherits MusicCd Class methods and attributes

**Class Attributes:**

public static final double FILE\_SIZE = 0.015

**Class Methods:**

public ClassicalGenre(Integer id, String title, Integer yearPublished, Double fileSize)

Call Parent Constructor with Parameters

public ClassicalGenre(String line)

Call Parent Constructor with Parameters

Call calculateFee()

public void calculateFee()

If yearPublished parameter equals Calendar Year

Set rentalFee equal to fileSize times FILE\_SIZE\_FEE plus CURRENT\_YEAR\_FEE

Else

Set rentalFee equal to fileSize times FILE\_SIZE\_FEE

**Class Name:** RomanceDvdGenre

**Class Description:** The RomanceDvdGenre class provides a descriptor for specific MovieDvd media types and Calculates Fees based on Genre specific requirements

**Class Modifiers:** Public

**Class Inheritance:** Inherits MovieDvd Class methods and attributes

**Class Attributes:**

public static final double ADDITIONAL\_FEE = 0.25

**Class Methods:**

public RomanceDvdGenre(Integer id, String title, Integer yearPublished)

Call Parent Constructor with Parameters

public RomanceDvdGenre(String line)

Call Parent Constructor with Parameters

Call calculateFee()

public void calculateFee()

If yearPublished parameter equals Calendar Year

Set rentalFee equal to CURRENT\_YEAR\_FEE plus ADDITIONAL\_FEE

Else

Set rentalFee equal to STANDARD\_FEE plus ADDITIONAL\_FEE

**Class Name:** DramaGenre

**Class Description:** The DramaGenre class provides a descriptor for specific MovieDvd media types and Calculates Fees based on Genre specific requirements

**Class Modifiers:** Public

**Class Inheritance:** Inherits MovieDvd Class methods and attributes

**Class Attributes:**

public static final double ADDITIONAL\_FEE = 0.25

**Class Methods:**

public DramaGenre(Integer id, String title, Integer yearPublished)

Call Parent Constructor with Parameters

public DramaGenre(String line)

Call Parent Constructor with Parameters

Call calculateFee(yearPublished)

public void calculateFee()

If yearPublished parameter equals Calendar Year

Set rentalFee equal to CURRENT\_YEAR\_FEE plus ADDITIONAL\_FEE

Else

Set rentalFee equal to STANDARD\_FEE plus ADDITIONAL\_FEE

**Class Name:** DocumentaryGenre

**Class Description:** The DocumentaryGenre class provides a descriptor for specific MovieDvd media types

**Class Modifiers:** Public

**Class Inheritance:** Inherits MovieDvd Class methods and attributes

**Class Methods:**

public DocumentaryGenre (Integer id, String title, Integer yearPublished)

Call Parent Constructor with Parameters

public DocumentaryGenre (String line)

Call Parent Constructor with Parameters

**Class Name:** <<Aspect>> LicenseFee

**Class Description:** The LicenseFee Aspect class will respond to the calculateFee() method execution, returning a calculated usageFee.

**Class Modifiers:** Public

**Class Inheritance:** None

**Class Attributes:**

public static final double FEE = 0.10

private integer tally

private double usageFee

**Class Methods:**

pointcut tallyCalculateFee()

Creates a join point at any call of calculateFee() method.

pointcut returnUsageFee()

Creates a join point at call of MediaRental.licenseFeeTotal() method.

advice before() tallyCalculateFee()

Increments Tally variable before tallyCalculateFee() join point.

advice around() returnUsageFee()

Returns Product of Tally and FEE value when returnUsageFee() join point called.

# 6 HUMAN INTERFACE DESIGN

## 6.1 Overview of User Interface

The User Interface will consist of a simplistic fixed window display where form fields are editable and provide greyed out tooltips. When the Attendant begins use of the Media Rental system they are offered two options, to either create a New Customer or to Open an existing Customer Account. This is shown in Figure 01.

Opening an existing Customer Account can be performed by entering a valid Customer ID in the Search query field next to the Customer Lookup label. Once an Existing Customer entry is located, the system will display User and Media data to the Attendant.

When creating a New Customer account, as seen in Figure 03, the attendant enters applicable User data in the Entry fields. At this point, the Attendant can either Enter New Media or can Save the User Account file.

At the Customer Entry display, the attendant can Save and Close, Add a New Rental, or Delete the Account. The New Rental dialog will popover existing elements of the user interface, disabling them. The Save and Close and Delete Account buttons will refresh the user interface to the state displayed in Figure 01.

## 

## 

## 6.2 Screen Images

A screenshot of a cell phone

Description generated with very high confidenceA screenshot of a cell phone

Description generated with very high confidenceA screenshot of a cell phone

Description generated with very high confidenceA screenshot of a cell phone

Description generated with very high confidence

## 6.3 Screen Objects and Actions

The New Rental Entry Popover Dialog will contain two drop down selection menus. These menus will modify the way that the Media Rental System calculates Rental Fees for specific Media and Genre types. The Genre Type dropdown selection menu will populate when a Media Type is selected in the dropdown selection menu. The system will auto calculate the rentalFee once the final Genre selection is made.

# 

# 

# 7 REQUIREMENTS MATRIX

|  |  |  |  |
| --- | --- | --- | --- |
| **Req. #:** | **Req. Desc.:** | **Classes** | **Methods** |
| R1 | System stores, retrieves, and allows editing of a singular User owned User Account File | MediaRental  UserAccount  User | addAccount()  openAccount()  saveAccount()  deleteAccount()  getSelectedAccount()  setSelectedAccount() |
| R2 | System stores, retrieves, and allows editing of zero to many Rented Media Data with associated User Account | UserAccount  Media | rentMedia()  returnMedia()  renewMedia() |
| R3 | System calculates Rental Fees for Movie DVD Price Structure | MovieDvd  RomanceDvdGenre  DramaGenre  DocumentaryGenre | calculateFee() |
| R4 | System calculates Rental Fees for Music CD Price Structure | MusicCd  CountryGenre  ClassicalGenre | calculateFee() |
| R5 | System calculates Rental Fees for Audio Book CD Price Structure | AudioBookCd  MemoirGenre  SciFiGenre  RomanceCdGenre | calculateFee() |
| R6 | System tracks each time the calculateFee() method is called for software licensing purposes. | <<Aspect>> LicenseFee  MediaRental | licenseFeeTotal() |