CS230 Design Document

Group 24

Group Members

Eduard Zakarian (965217)

Luke Burtonwood (957005)

Sze Chai Wong (690463)

Nathan Bowen (911142)

Sai Sudharshan Govindarajan (956299)

Joshua Placidi (910252)

Isaac McIntyre (951975)

November 03, 2018

Contents

1 Candidate Classes and Responsibilities……3

1.1 Resource classes……..…….3

1.2 Avatar classes……...……..4

1.3 Fine and DataAdapter classes……..…..4

1.4 Account classes…….....5

1.5 Graphic classes……...6

2 Class Diagrams…….…..7

2.1 Resource classes……7

2.2 Avatar classes…….….9

2.3 Fine and DataAdapter classes…….….9

2.4 Account classes…….....10

2.5 Graphic classes……..11

3 Hierarchy Descriptions……....12

4 Collaboration Descriptions……14

5 Operation Descriptions……....15

5.1 DataAdapter writer (DataAdapter class).....15

5.2 DataAdapter reader (DataAdapter class).....15

5.3 Generating a unique ID (Resource class).....15

5.4 Custom Drawer (CustomDrawing class).....15

5.5 Fine Operation (Fine Class).....15

Chapter 1: Candidate Classes and Responsibilities

1.1 Resource classes

Resource
A template for creating either a kind of Book, Laptop or a DVD
Author: Sze Chai Wong
SuperClass: -
SubClasses: Copy
Responsibilities: • Holds all general information that applies to all resource subclasses, such as unique ID, year and a profile image
Collaborations: Avatar, CustomDrawing, DataAdapter, User

Book	Laptop
Stores information specific to one exact instance of book	Stores information specific to one exact instance of laptop
Author: Sze Chai Wong	Author: Luke Burtonwood
SuperClass: Copy	SuperClass: Copy
SubClasses: -	SubClasses: -
Responsibilities: • Sets and gets information about the book • Create an object of a book	Responsibilities: • Sets and gets information about the laptop • Create an object of a laptop
Collaborations: -	Collaborations: -

DVD	Copy
Stores information specific to one exact instance of a DVD	An actual copy of a Book, Laptop or a DVD
Author: Luke Burtonwood	Author: Eduard Zakarian
Superclass: Copy	Superclass: Resource
Subclasses: -	Subclasses: Book, DVD, Laptop
Responsibilities: • Sets and gets information about the book • Create an object of a DVD	Responsibilities: • Sets and gets information about the actual copy of a resource • Create a copy of a resource • Stores loan duration and a date this copy is due
Collaborations: -	Collaborations: User

1.2 Avatar classes

Avatar	CustomDrawing
Hold a locally saved image which may be used as user's profile picture	Provides functionality to create a custom avatar
Author: Nathan Bowen	Author: Nathan Bowen
SuperClass: -	SuperClass: -
SubClasses: -	SubClasses: -
Responsibilities: • Create an avatar object • Set and get a path to an image	Responsibilities: • Provide an environment to draw an avatar • Ability to draw straight lines • Ability to draw a particle trace
Collaborations: User	Collaborations: User

1.3 Fine and DataAdapter class

Fine	DataAdapter
Hold a description of a penalty and the amount of money the user might need to pay	Convert SQL queries into class objects and vice versa
Author: Sai Sudharshan Govindarajan	Author: Eduard Zakarian
SuperClass: -	SuperClass: -
SubClasses: -	SubClasses: -
Responsibilities: • Create an object of Fine • Apply a penalty to a User • Set the price and description of a penalty	Responsibilities: • Populate the database by converting objects to SQL queries • Return an object when processing a given query
Collaborations: User, LibrarianPage, UserPage	Collaborations: Resource, Book, DVD, Laptop, User, Dashboard, LibrarianPage, UserPage, LoginPage, InventoryBrowser, Homepage

1.4 Account classes

Account	Librarian
Holds general account information about the user	Holds librarian specific information such as staff number and employment date. Also, a Librarian has the permission to issue books to borrowers and accept returned items
Author: Josh Placidi	Author: Josh Placidi
SuperClass: -	SuperClass: Account
SubClasses: Librarian, User	SubClasses: -
Responsibilities: • A template for creating either a User or a Librarian • Set and get user's data	Responsibilities: • Create Librarian object • Sets and gets librarian specific data • Create new resources • Edit existing resources • Loan copies to borrowers • Process returned copies • Issue fines • Pay off fines
Collaborations: Librarian, User, Fine, DataAdapter	Collaborations: Fine

User
Holds information, specific to a user such as the list of borrowed items and a flag which is triggered when an individual has unpaid fines
Author: Josh Placidi
SuperClass: Account
SubClasses: -
Responsibilities: • Create User objects • Sets and gets user's specific data • Tracks if a user has any unpaid fines • Tracks if a user has any overdue copies • Tracks the list of borrowed resources • Return a copy to a library • Keeping track of a balance
Collaborations: Resource, Fine, UserPage

1.5 Graphic classes

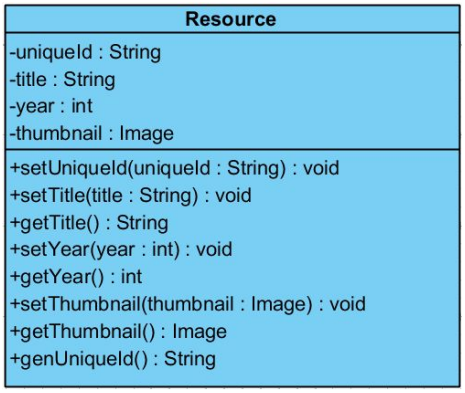
Dashboard	HomePage
Call the other dashboard classes.	A home page for user in library to visit.
Author: Sze Chai Wong	Author: Sze Chai Wong
SuperClass: -	SuperClass: Dashboard
SubClasses: HomePage, InventoryBrowser, LoginPage, LibrarianPage, UserPage	SubClasses: -
Responsibilities: • Switching between different User Interface	Responsibilities: • Provide an interface for user to choose to browse the inventory or login
Collaborations: DataAdapter	Collaborations: DataAdapter, InventoryBrowser

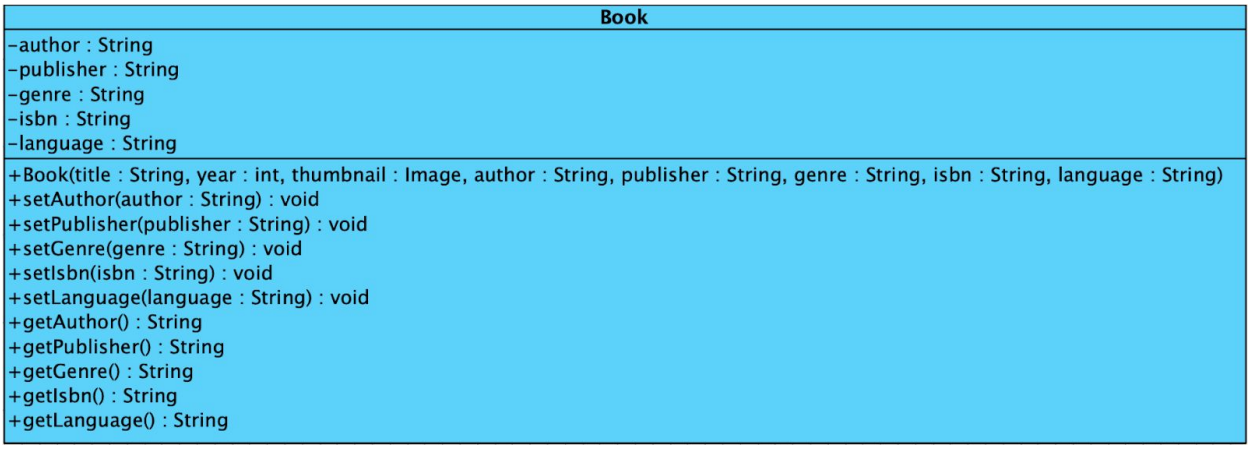
InventoryBrowser	LoginPage
An Interface for a user to browse the resource in the library.	A page for user/librarian to login.
Author: Sze Chai Wong	Author: Sze Chai Wong
SuperClass: Dashboard	SuperClass: Dashboard
SubClasses: -	SubClasses: -
Responsibilities: • Provide filters for user to search for specific resource	Responsibilities: • Provide 2 input space for user/librarian to type their username and password.
Collaborations: DataAdapter, LibrarianPage, UserPage, HomePage	Collaborations: DataAdapter, LibrarianPage, UserPage

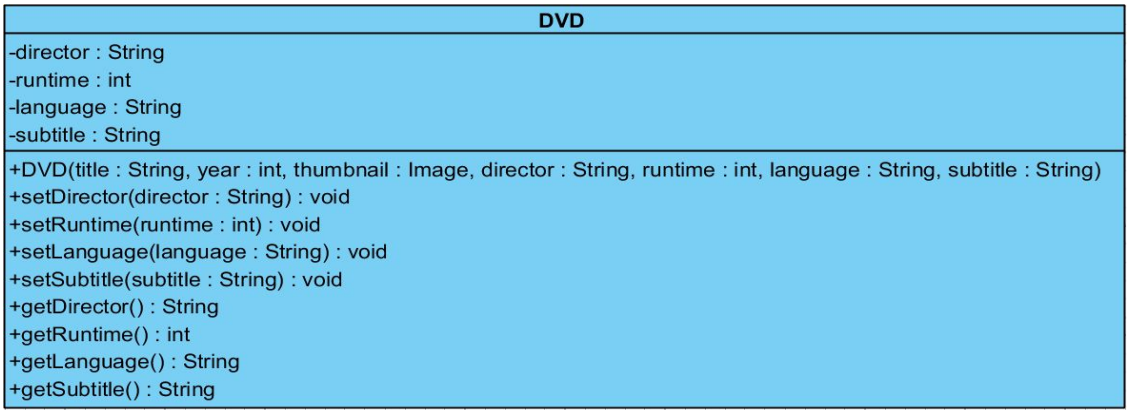
LibrarianPage	UserPage
A page for librarian.	A page for user.
Author: Sze Chai Wong	Author: Sze Chai Wong
SuperClass: Dashboard	SuperClass: Dashboard
SubClasses: -	SubClasses: -
Responsibilities: • Able to create/edit user or resource • Help user to borrow/return items • Manage fine payments • View borrowed/overdue resources	Responsibilities: • Notify user the number of overdue items he/she has • View borrowed/requested/reserved items • View his/her own transaction history • View his/her own balance
Collaborations: DataAdapter, Fine, InventoryBrowser, LoginPage	Collaborations: DataAdapter, Fine, InventoryBrowser, LoginPage, User

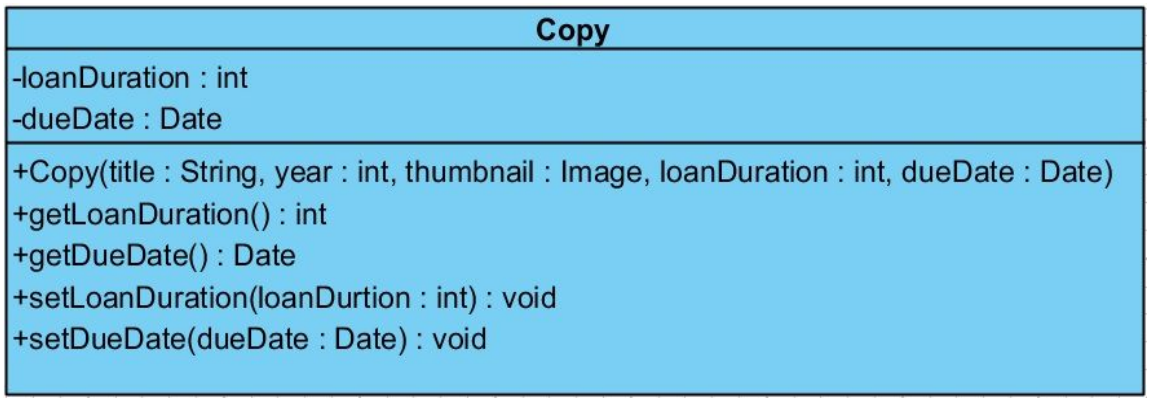
Chapter 2: Class diagrams

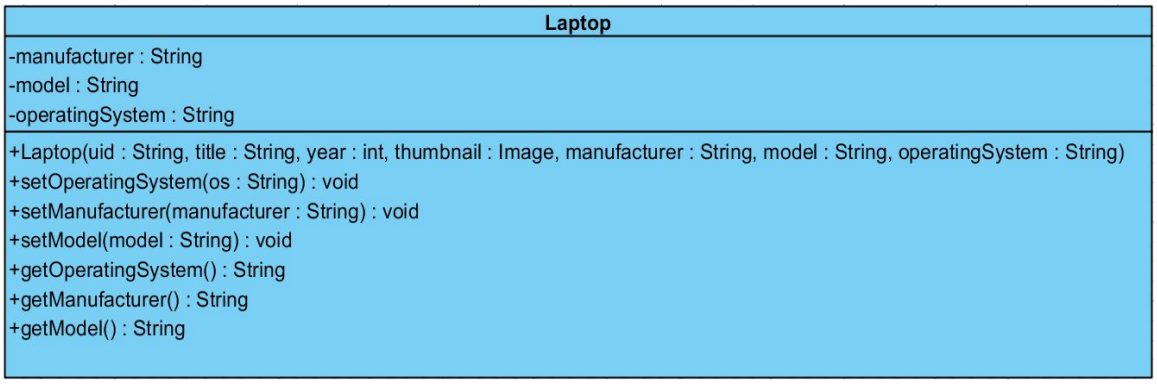
2.1 Resource classes



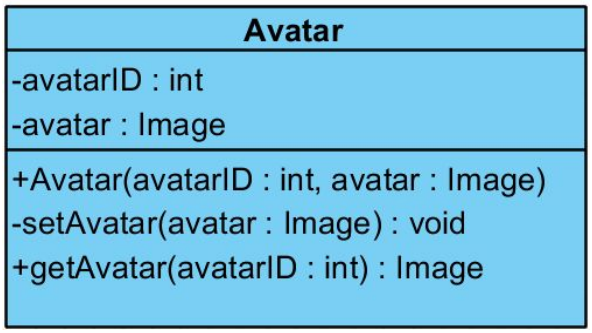


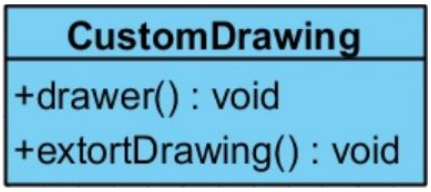

DVD
-director : String
-runtime : int
-language : String
-subtitle : String
+DVD(title : String, year : int, thumbnail : Image, director : String, runtime : int, language : String, subtitle : String)
+setDirector(director : String) : void
+setRuntime(runtime : int) : void
+setLanguage(language : String) : void
+setSubtitle(subtitle : String) : void
+getDirector() : String
+getRuntime() : int
+getLanguage() : String
+getSubtitle() : String


Copy
-loanDuration : int
-dueDate : Date
+Copy(title : String, year : int, thumbnail : Image, loanDuration : int, dueDate : Date)
+getLoanDuration() : int
+getDueDate() : Date
+setLoanDuration(loanDurtion : int) : void
+setDueDate(dueDate : Date) : void

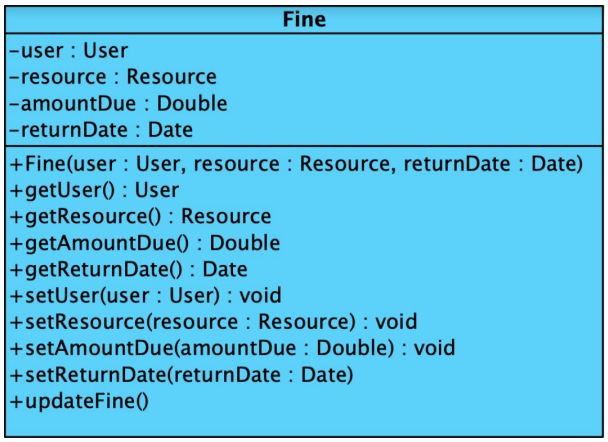

Laptop
-manufacturer : String
-model : String
-operatingSystem : String
+Laptop(uid : String, title : String, year : int, thumbnail : Image, manufacturer : String, model : String, operatingSystem : String)
+setOperatingSystem(os : String) : void
+setManufacturer(manufacturer : String) : void
+setModel(model : String) : void
+getOperatingSystem() : String
+getManufacturer() : String
+getModel() : String

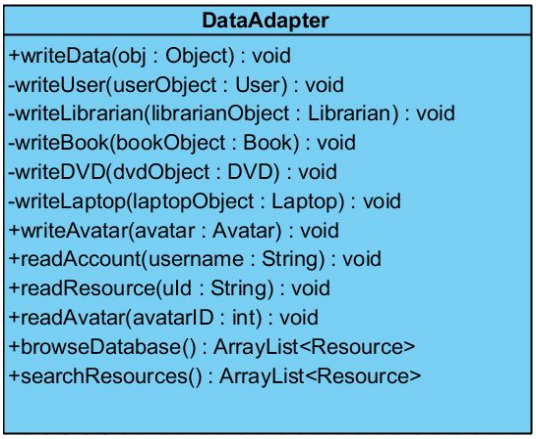
2.2 Avatar classes





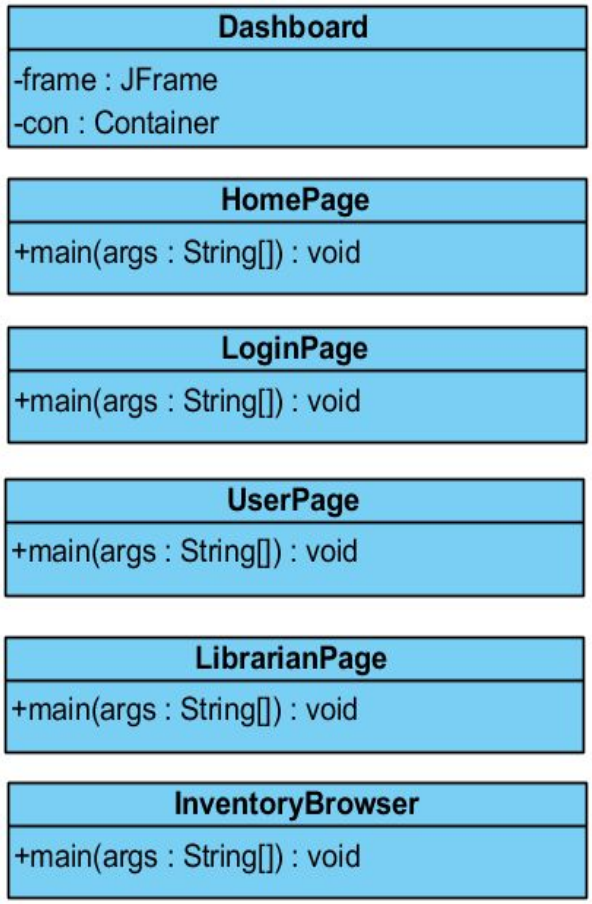
2.3 Fine and DataAdapter classes



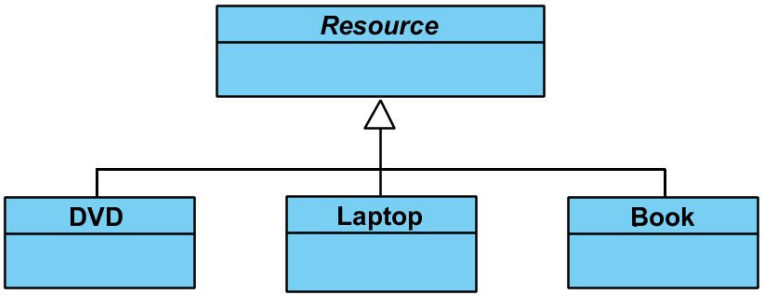


2.4 Account classes

2.5 Graphic classes

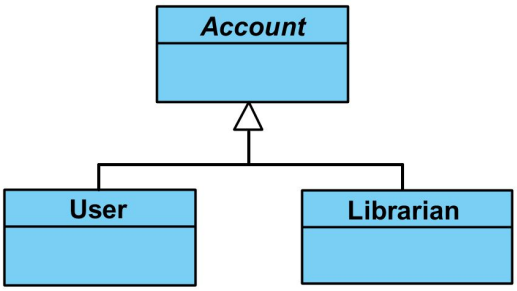


Chapter 3: Hierarchy Descriptions



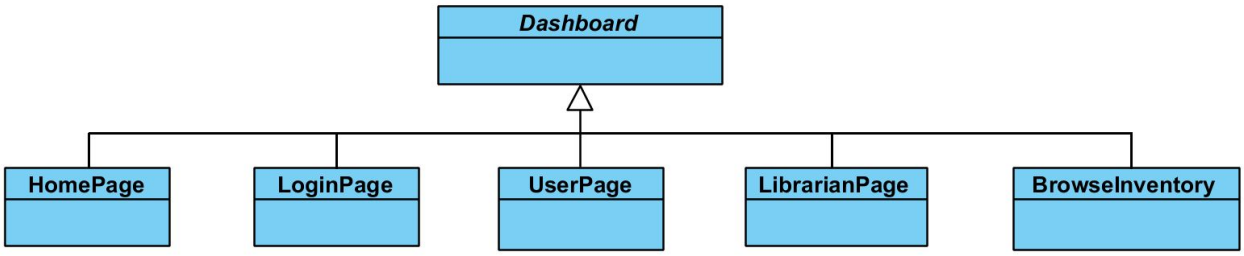
Resource -> (Book, DVD, Laptop)

Book, DVD and Laptop are kind of resources, in which they all inherit UniqueID, Title of the resource, Year resource created and a thumbnail of that resource. Resource is the abstract.



Account -> (User, Librarian)

User and Librarian are kind of Accounts, in which they both inherit username of account, name of account owner, phone number of account owner, address of account owner and a custom avatar chosen by the account owner. Account is the abstract.

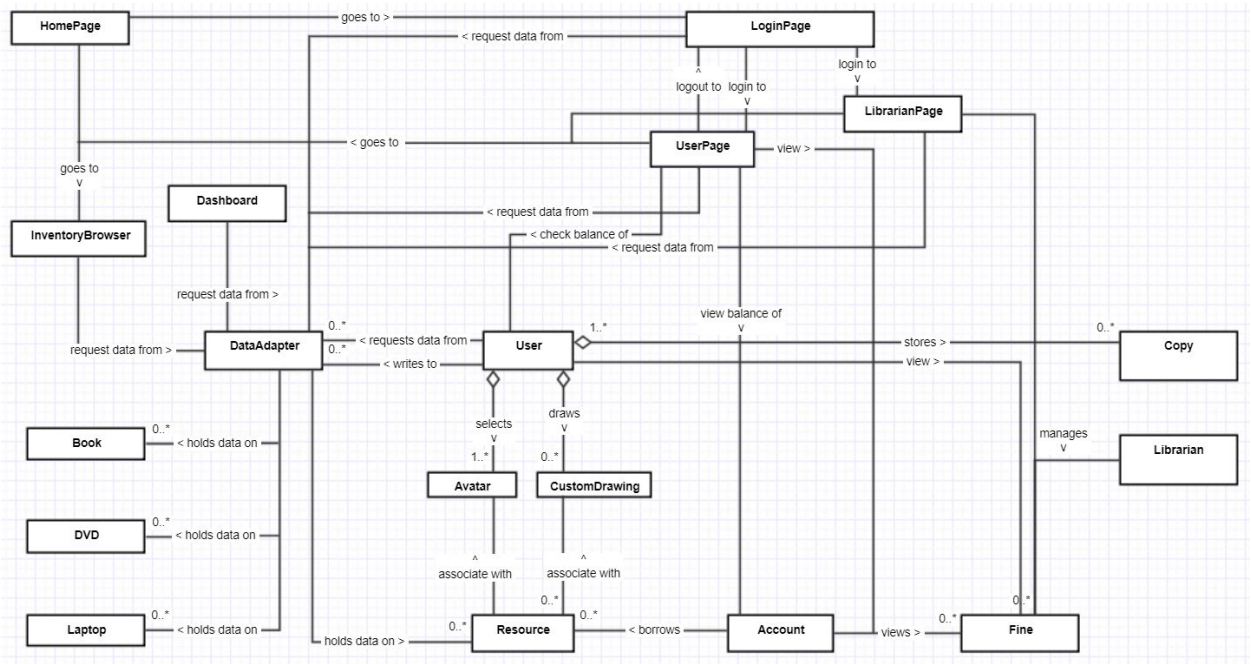


Dashboard ->

(HomePage, LoginPage, UserPage, LibrarianPage, InventoryBrowser)

HomePage, LoginPage, UserPage, LibrarianPage, InventoryBrowser are kind of dashboard, in which they all inherit JFrame and Container from Dashboard. Dashboard is the abstract.

Chapter 4: Collaboration Descriptions



Aggregations

User is made up of Avatar and User is made up of CustomDrawing - as each User can have an Avatar or CustomDrawing to make up the User's profile information. The user must have either one or the other, so it does not own both of them.

User is made up of Copy as the User will store an arraylist of copies of resources that they are borrowing to keep track of specific copies of each Resource that has been borrowed.

This design for the project consists of a group of large classes, with not many combining together to split information up. Therefore, there is no composition relationships between the classes.

Chapter 5: Operation Descriptions

5.1 Writing to the database (DataAdapter class)

To write (insert) something to the database, the writeData() method should be used. It takes any object as a parameter and then calls one of the private methods, namely writeUser(), writeLibrarian(), writeBook(), writeDVD(), writeLaptop(), writeAvatar(). Each of these private methods will take an object as a parameter, gather all the information and write it to the database using the SQL query, which will also be generated in the method.

5.2 Reading from the database (DataAdapter class)

To read something from the database we will use a set of public methods, readAccount() takes a unique username as a parameter and gathers all the information about that particular account from the database. This method returns an object of a type Account. readResource() takes resources unique ID as a parameter and gathers all the information about that particular resource from the database. This method returns an object of a type Resource

5.3 Generating a unique ID (Resource class)

For each resource, a unique ID will be generated and stored in the database, along side that resource. To do this, we will make a method to search for the last resource, using the DataAdapter class, of the type created, which will be Book, DVD, or Laptop. The 3 different resources will have a unique letter at the start of the ID to show it is the certain resource, Book will have a “B”, DVD will have a “D” and Laptop will have an “L”. It will then find the last created Unique ID and create an ID incremented from the last. There will also be a method to check there is no duplicate uniqueIDs in case of error in the database.

5.4 Custom Drawer (CustomDrawing Class)

The CustomDrawer operations primary purpose will be to allow users to create and draw an image, which can be exported and used as an avatar, which is selected in the avatar class. We considered a variety of different options on how this could be implemented, such as using swing gui functions, but have decided JavaFX features would better suit our program. JavaFX canvas gives us the ability to setup a blank window, which will be used as the entire drawing space, where multiple different properties such as drawing lines and shapes can easily be implemented, using the GraphicsContext function inside JavaFX. We plan on implementing the ability to draw straight lines, and draw a particle trace, with a variety of colour choices and brush stroke sizes available to choose from. This can all be done inside a singular method, which we have named ‘drawer’, by adding properties using GraphicsContext. This also gives us the opportunity, to implement additional features such as filling shapes and applying effects, which will be looked at further during the implementation stage.

5.5 Fine Operation (Fine Class)

A fine is applied to a user if they do not return a book by the set return date. When a fine is issued a new fine object is created with the given user, given resource and return date passed into the constructor. Every active fine a user has is stored in the database. To pay off a fine the user must visit a librarian, the librarian is able to see information about all fines a user has through a database query. The user can decide which fines they would like to pay off, any fines they do not pay fully will continue to increase per the daily resource rate. The payFine() method in the librarian class will be called whenever a user pays a fine, it will first call the updateFine() method which will update the amount due depending on the date, then reduce the fine amount by the amount the user pays. If a user has any outstanding fines then they will be prevented from borrowing a resource from the library until all fines are paid.