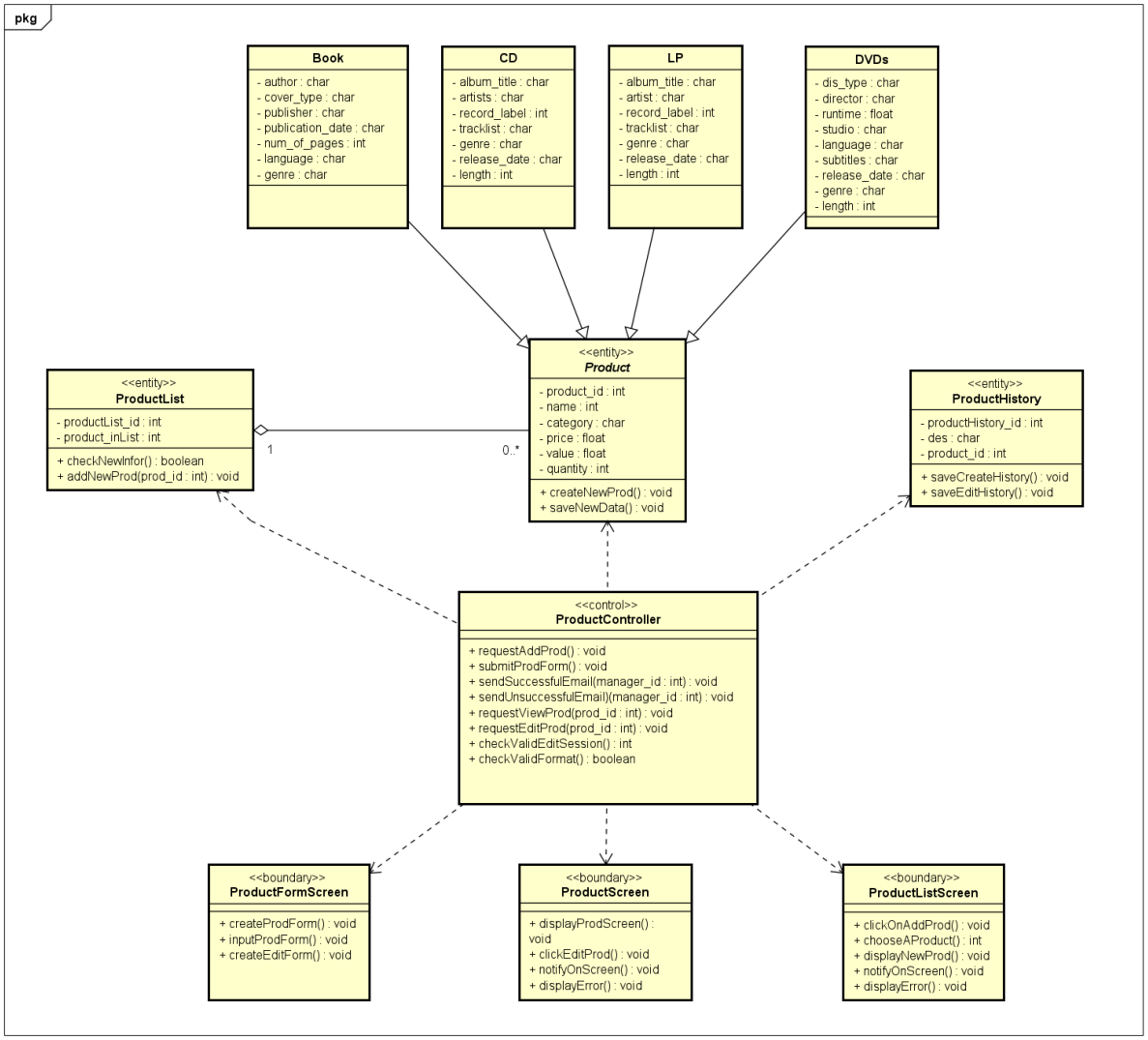
Use Case “Add/Edit a product”

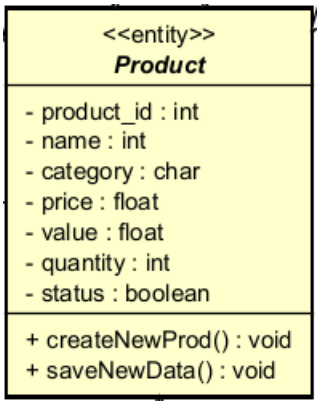
1. Class Design
   1.1. Overall



   1.2. Detail class design

      1.2.1. Product Class

      Product is an abstract class that helps AIMS software define a common base about all products, since each category of products has some unique fields.



Attribute Table

#	Name	Data Type	Default Value	Description
1	product_id	int		An integer represents a unique identifier for the product
2	name	string		A string represents the name or title of the product
3	category	string		A string represents the classification of the product
4	price	float		A float represents the selling price of the product
5	value	float		A float represents the base value of the product
6	quantity	int		An integer represent the number of copies in stock
7	status	boolean	false	True: product is displayed on store False: product is hidden on store

Operation Table

#	Name	Return Type	Purpose
1	createNewProd	void	Create a new product after the product manager requests to add a product. This is a constructor
2	saveNewData	void	Save edited information of a product when product manager requests to update new information for a specific product.

Parameters:

Exceptions: No exceptions since data was checked before calling this operation

Methods:

saveNewData(): void

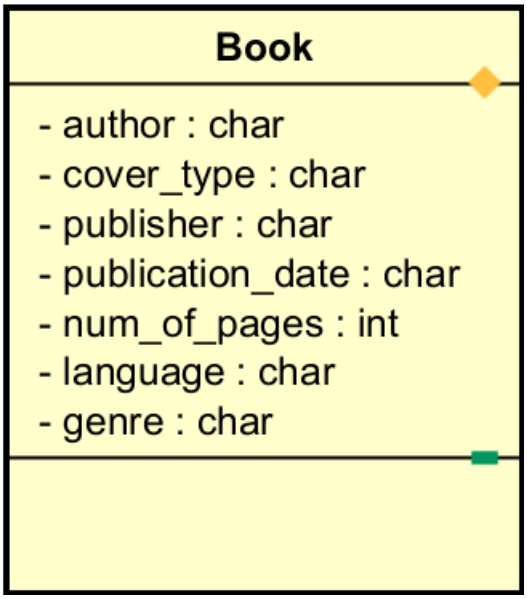
- Flow: updates edited data for a product
- Return type: void

How to use:

- Both of the following operations are automatically triggered during UC "Add a Product" and "Edit a Product"
- The "status" attribute in Product class is provide due to: Appear/Hidden status help Prod Manager control their shop if an item is out of stock

This abstract class currently has 4 extended class names: CD, DVDs, LP, Book.

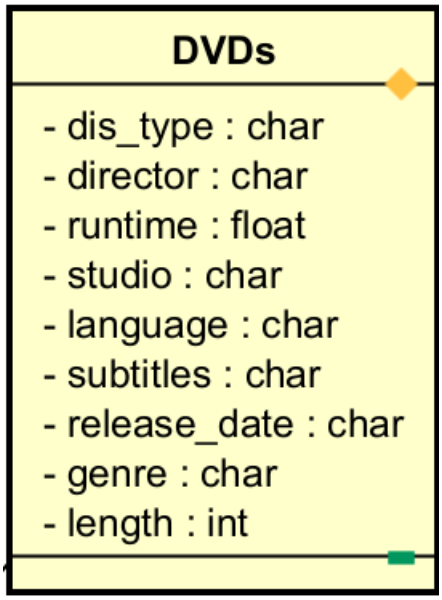
1.2.1.1. Book Class



#	Name	Data Type	Default Value	Description
1	author	char		A string represents the name of this book ‘s author(s)
2	cover_type	char		A string represents the cover type of this book
3	publisher	char		A string represents the name of this book ‘s publisher
4	publication_date	char		A string represents the public date
5	num_of_pages	int		An integer represent the number of pages in this book
6	language	char		A string represents language(s)

				used in this book
7	genre	char		A string represents genre(s) of this book

1.2.1.2. DVDs Class

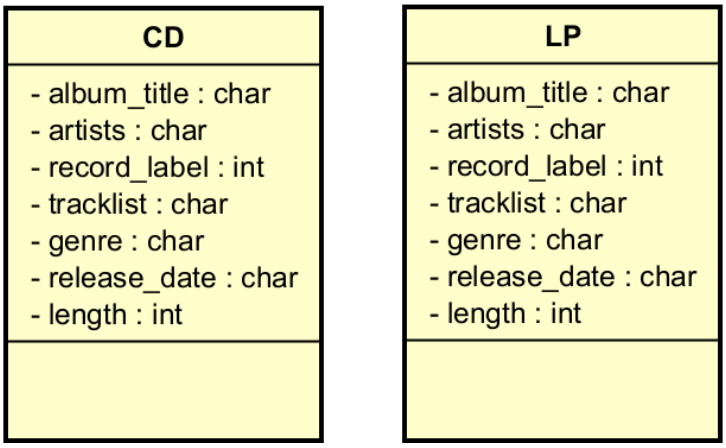


#	Name	Data Type	Default Value	Description
1	dis_type	char		A string represents the type of this DVD
2	director	char		A string represents the director of this DVD
3	runtime	float		A float represents the maximum amount of video playback time it can hold

4	studio	char		A string represents the studio name of this DVD
5	language	int		A string represents language(s) used in this DVD
6	subtitles	char		A string represents subtitles language(s) used in this DVD
7	release_date	char		A string represents the release date of this DVD
8	genre	char		A string represents genre(s) of this DVD
9	length	int		An integer represents the length in second of this DVD

1.2.1.3. LP and CD Class

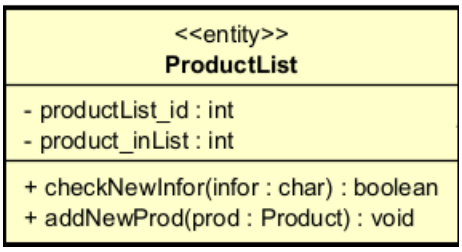
These two classes have the same attributes. Of course, we can merge both into a class and add one attribute to define the type, like “isBook: boolean”. However, dividing into 2 separate classes may help decrease the query time of AIMS systems.



#	Name	Data Type	Default Value	Description
1	album_title	char		A string represents the title of album CD/LP belong to
2	artist	char		A string represents name of the artist(s)
3	record_label	intt		A float represents the record label of this CD/LP
4	tracklist	char		A string represents the name of track list of this CD/LP
5	release_date	char		A string represents the release date of this CD/LP
6	genre	char		A string represents genre(s) of this CD/LP
7	length	int		An integer represents

				the length in second of this CD/LP

1.2.2. ProductListClass



Attribute Table

#	Name	Data Type	Default Value	Description
1	productList_id	int		An integer represents a unique identifier for the productList (aka store_ID)
2	Product_inList	int		A reference to a product_ id in product database

Operation Table

#	Name	Data Type	Description
1	checkNewInfor	boolean	An operation can check if edit data is invalid or not (new data must be unique at some fields in comparison with existing product in store)
2	addNewPro	void	Add reference to a product_ id in product database. This makes product manager ‘store have one more product.

Parameters:

- *addNewProd (prod_id: int): void* : product_id is argurment to add a specific product to product list.

Exceptions:

- checkNewInfo is a check operation used in UC "Add a product "and "Edit a product" so we can throw exception due to reject reason: newDataExisted
- No exceptions since valid data was automatically fill in when calling *addNewProd* operation.

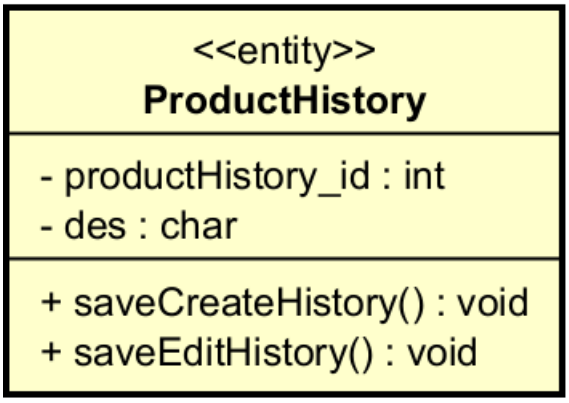
However, if we have an UC like "Add an product to store", *addNewProd (prod_id: int): void should* throw invalidIdException when prod_id arguments cannot be found, be negative or zero.

Methods:

- checkNewInfo(): boolean:
  Flow:
  - Doing a while loop through all copies in Product List class to check if there is product currently in product manager product list has same name as new add/edit product.
  - Return true if no duplicate value, else return false and throw exception
- addNewProd (prod_id: int): void :
  Flow:
  - In UC "Add a product", after creating a new product and add to product class. AIMS software uses this method to add that product to product list. Meaning display it in product manager store.

How to use: both of the following operations are automatically triggered during UC "Add a Product" and "Edit a Product"

1.2.3. ProductHistory Class



Attribute Table

#	Name	Data Type	Default Value	Description
1	productHistory_id	int		An integer represents a unique identifier for the productHistory
2	product_id	int		A reference with product with its history
3	des	char		A string represents description about all add/ edit history

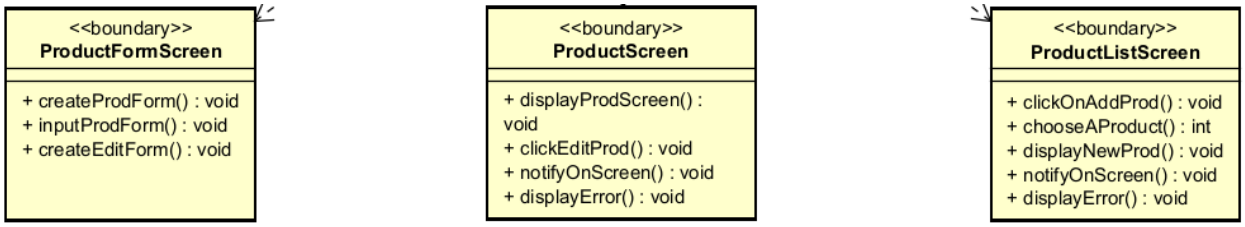
Operation Table

#	Name	Data Type	Description
1	saveCreateHistory	void	Construction of productHistory
2	saveEditHistory	void	Save edit session of a product

Parameters: No parameters
Exceptions: No exception
Methods:

- checkNewInfo(): boolean:
  Flow:
  - After create a product successfully, AIMS software call this method to save the create session with description: "[product_id] : is created at [time] "
- addNewProd (prod_id: int): void :
  Flow:
  - After edit a product successfully, AIMS software call this method to save the create session with description: "[product_id] : is edit at [time] , [value x] to [value y] "

1.2.4. Boundary Classes



Product Form Screen Table

#	Name	Data Type	Description
1	createProdForm	void	Create a Product Form and display it for product manager fill in when call UC "Add a Product"
2	Input Product Form	void	Receive data inputted by product manager
3	createEditForm	void	Create a Product Form and display it for product manager fill in when call UC "Edit a Product"

Parameters: No parameters
Exceptions: No exception
Methods:

- createProdForm():void :
  Flow:
  - AIMS software create a Product Form for product manager fill in when call UC "Add a Product"

- inputProdForm():void
  Flow:
  - Receive data input by product manager
- createEditForm():void:
  Flow:
  - -Create a Product Form for product manager fill in when call UC "Edit a Product"

Product Screen Table

#	Name	Data Type	Description
1	displayProdScreen	void	Display product detail information on screen

2	clickEdtiProd	void	Trigger when user clicks on Edit Button on screen
3	notifyOnScreen	void	Notify status of an operation on screen. (In this case, it belonged to Add/Edit a Product.)
4	displayError	void	Notify reason leads to Error of an operation on screen. (In this case, it belonged to Add/Edit a Product.)

Parameters: No parameters
Exceptions: No exception
Methods:

- displayProdScreen() : void:
  Flow:
  - AIMS software display product detail information on screen after product manager click on a product on screen.

- clickEditProd() : void
  Flow:
  - Trigger when user clicks on Edit Button on screen
  - Product controller checks session then creates an Edit Form or display error (attempts/day > 30)
- notifyOnScreen() : void
  Flow:
  - Notify status of an operation on screen. (In this case, it belonged to Add/Edit a Product.)
- displayError() : void:
  Flow:
  - Notify reason leads to Error of an operation on screen. (In this case, it belonged to Add/Edit a Product.)

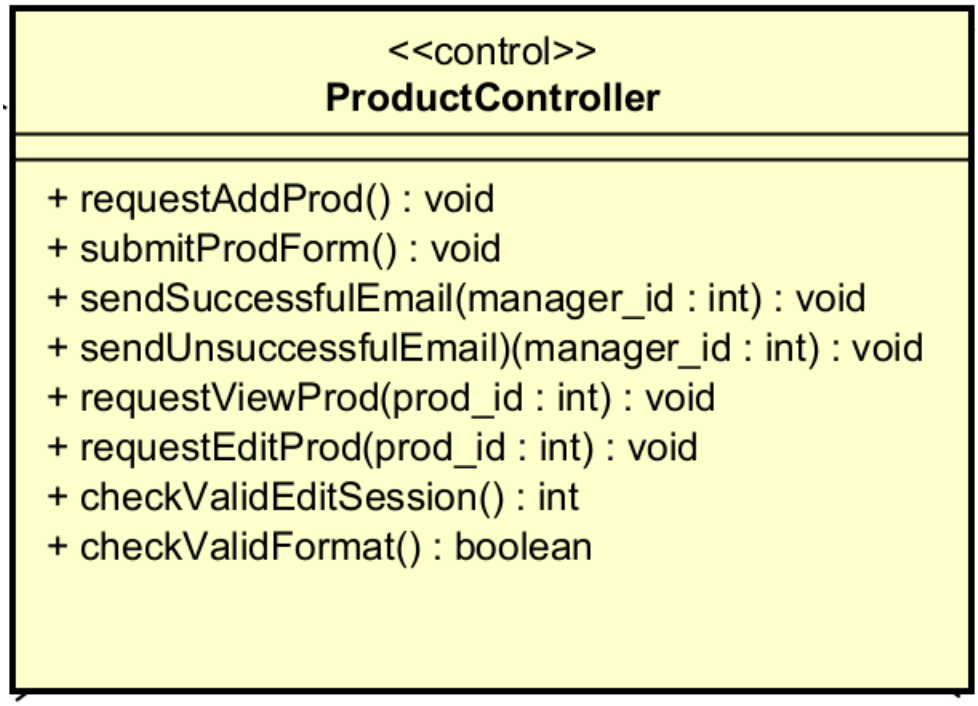
Product List Screen Table

#	Name	Data Type	Description
1	clickOnAddProd	void	Trigger when user clicks on a Add a Product on Product List Screen. Call UC “Add a Product”

2	chooseAProduct	void	Trigger when user clicks on a product on product list screen. This also request controller to display product screen
3	displayNewProd	void	Display new product on shop
4	notifyOnScreen	void	Notify reason leads to Error of an operation on screen. (In this case, it belonged to Add/Edit a Product.)
5	displayError	void	Notify reason leads to Error of an operation on screen. (In this case, it belonged to Add/Edit a Product.)

Parameters: No parameters
Exceptions: No exception
Method: Flow as same as description
How to use: Detail is in Description Column in table

1.2.5. Product Controller Class



Product List Screen Table

#	Name	Data Type	Description
1	requestAddProd	void	Request to add a product
2	submitProdForm	void	Submit data from Product Form
3	sendSuccessfulEmail	void	Send successful email to user
4	sendUnsuccessfulEmail	void	Send unsuccessful email to user
5	requestViewProd	void	Request to view a product detail
6	requestEditProd	void	Request to Edit a Product
7	checkValidEditSession	int	Return number of editing attempts per day
8	checkValidFormat	void	Check if data input is in valid format

Parameters:

- checkValidEditSession (user_id/product_id: int): int (de bai khong noi ro 1 nguoi hay 1 san pham khong duoc edit qua 30 lan 1 ngay)
- sendSuccessfulEmail(user_id: int):void
- sendUnsuccessfulEmail(user_id: int):void
- requestEditProd(prod_id: int):void
- requestViewProd(prod_id: int):void

Exceptions: No exception because all ID is taken automatically based on click on UI and user login session.

Method & how to use: Detail is this table

#	Name	Flow/How to Use	Return Value
1	requestAddProd():void	The method is used after product manager clicks on Add a Product Button on ProductList Screen.	

		Controller creates a product form as sequence diagram	
2	submitProdForm():void	Method is used after product manager click on Add a Product Button Input data then be checked for validity	
3	sendSuccessfulEmail(user_id: int):void		
4	sendUnsuccessfulEmail(user_id: int):void		
5	requestViewProd(prod_id: int):void		
6	requestEditProd(prod_id: int):void	The method is used after product manager clicks on Edit Button on Product Screen. Controller creates a product form as sequence diagram	
7	checkValidEditSession(): int	Return number of editing attempts per day	Attemps is return value used to check valid session. If attempt < 30, product manager can continue, else display error

8	checkValidFormat():boolean	Check if data input is in valid format	If return value is true: the flow continues, else request user to input again