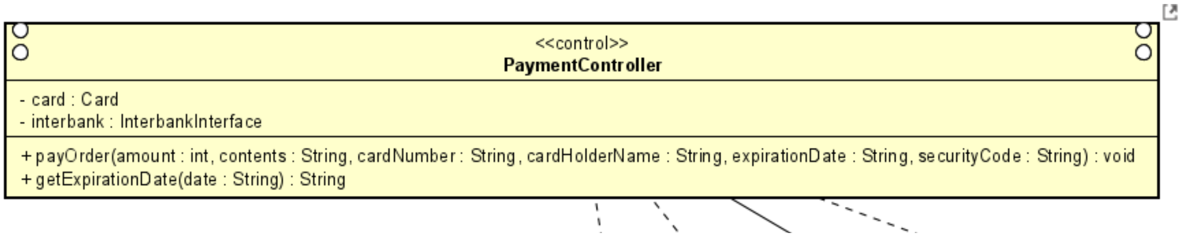
Programming Documentation

1. Class “PaymentController”



<<control>>
PaymentController
- card : Card
- interbank : InterbankInterface
+ payOrder(amount : int, contents : String, cardNumber : String, cardHolderName : String, expirationDate : String, securityCode : String) : void
+ getExpirationDate(date : String) : String

Attribute

#	Name	Data type	Default value	Description
1	card	Card	NULL	Represent the card used for payment
2	interbank	InterbankInterface	NULL	Represent the Interbank subsystem

Operation

#	Name	Return type	Description (purpose)
1	payOrder	Map<String, String>	Pay order, and then return the result with amessage

Parameter:

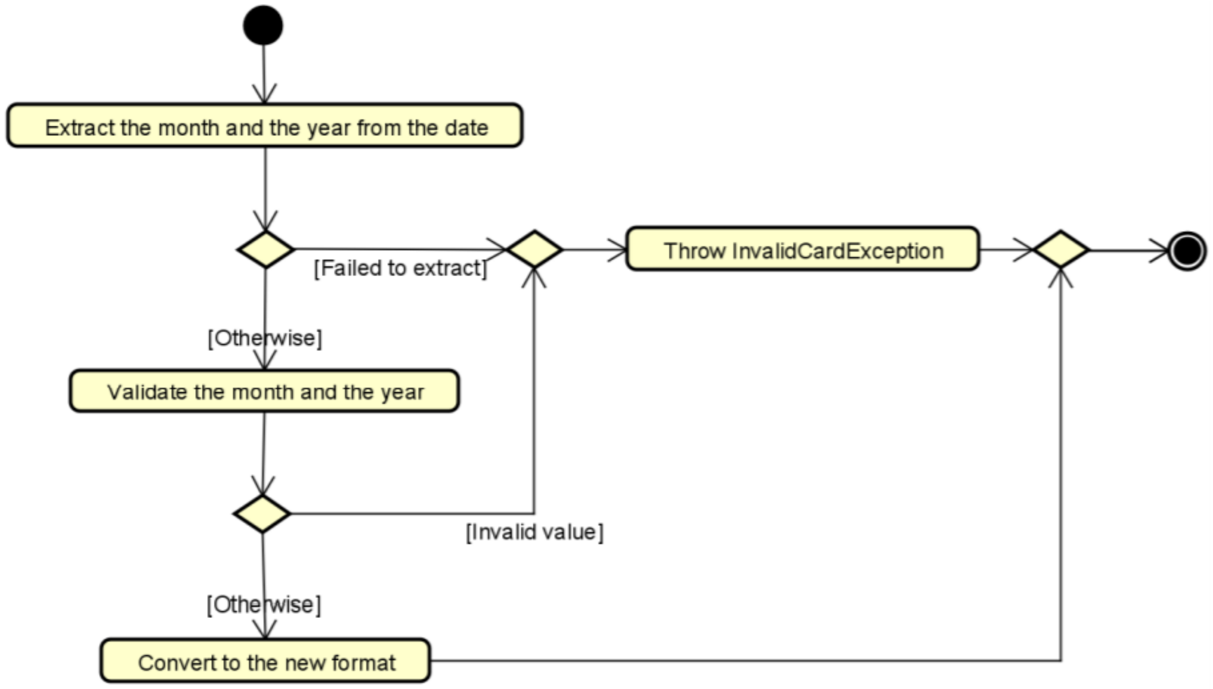
- amount - the amount to pay
- contents - the transaction contents
- cardNumber - the card number
- cardHolderName - the card holder name
- expirationDate - the expiration date in the format "mm/yy"
- securityCode - the cvv/cvc code of the credit card

Exception:

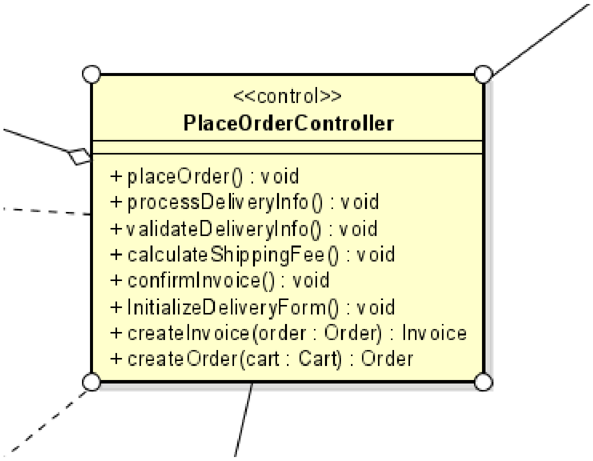
- None

Method

getExpirationDate: Given the String “date” representing the expiration datein the format "mm/yy", this method convert it into the required format “mmyy.”


Extract the month and the year from the date
[Failed to extract]
Throw InvalidCardException
[Otherwise]
Validate the month and the year
[Invalid value]
[Otherwise]
Convert to the new format

2. Class “PlaceOrderController”


<<control>>
PlaceOrderController
+placeOrder() : void
+processDeliveryInfo() : void
+validateDeliveryInfo() : void
+calculateShippingFee() : void
+confirmInvoice() : void
+InitializeDeliveryForm() : void
+createInvoice(order : Order) : Invoice
+createOrder(cart : Cart) : Order

Attribute

- None

Operation

#	Name	Return type	Description (purpose)
1	placeOrder	void	Place an order, receiving the entity Cart

2	processDeliveryInfor	Order	Represent DeliveryForm, store the information of User, Utils, and Cart and then export to an order
3	validateDeliveryInfo	boolean	Validating the Delivery information, receiving the entity Order
4	calculateShippingFees	void	Calculating the sum of Media and the shipping fees, receiving the entity Order
5	confimInvoice	void	Confirming the Invoice, receiving the entity Invoice
6	InitializeDeliveryForm	void	Create a new Delivery Form
7	createOrder	Order	Create a new order
8	createInvoice	Invoice	Create a new Invoice

Parameter:

- None

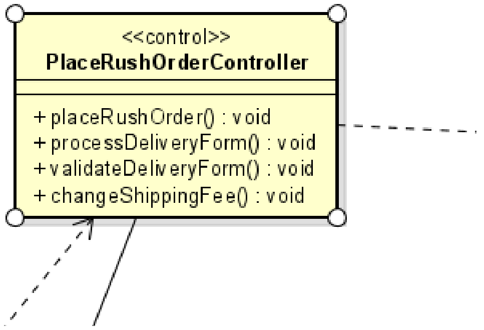
Exception:

- None

Method

- None

3. **Class "PlaceRushOrderController"**


<<control>>
PlaceRushOrderController
+ placeRushOrder() : void
+ processDeliveryForm() : void
+ validateDeliveryForm() : void
+ changeShippingFee() : void

Attribute

- None

Operation

#	Name	Return type	Description (purpose)
1	changeShippingFees	void	Changing the fomula to calculate the shipping fees when user uses rush order
2	placeRushOrder	void	Operating the action of Place Rush Order, check the valid Media and Location of Delivery, receiving the entity Order
3	validateDeliveryForm	boolean	Validating the delivery form when the customer desires to place a rush order
4	processDeliveryForm	void	Processing the delivery form after customer submits it

Parameter:

- None

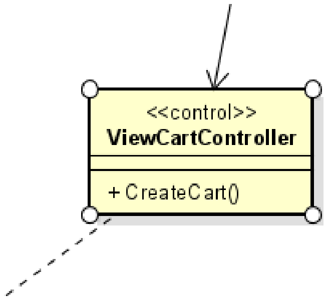
Exception:

- None

Method

- None

4. Class “ViewCartController”



Attribute

- None

Operation

#	Name	Return type	Description (purpose)

1	createCart	Cart	Creating the cart, which has the list of Media

Parameter:

- None

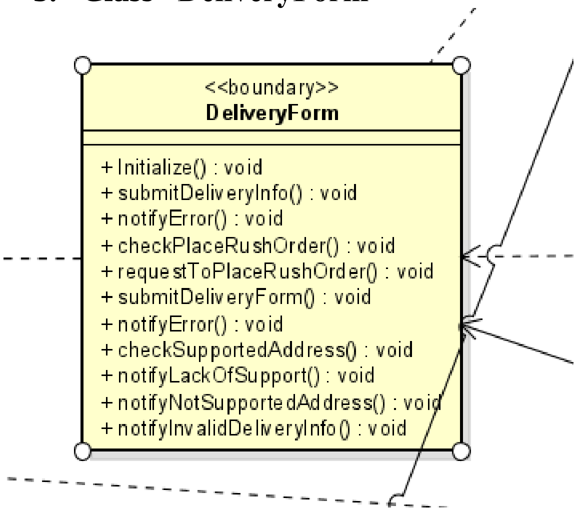
Exception:

- None

Method

- None

5. Class “DeliveryForm”


<<boundary>>
DeliveryForm
+ Initialize() : void
+ submitDeliveryInfo() : void
+ notifyError() : void
+ checkPlaceRushOrder() : void
+ requestToPlaceRushOrder() : void
+ submitDeliveryForm() : void
+ notifyError() : void
+ checkSupportedAddress() : void
+ notifyLackOfSupport() : void
+ notifyNotSupportedAddress() : void
+ notifyInvalidDeliveryInfo() : void

Attribute

- None

Operation

#	Name	Return type	Description (purpose)
1	initialize	void	Initializing the Delivery Form after clicking the button “Place Order” in class ViewCartScreen, receiving the entity Cart
2	submitDeliveryInfo	void	Saving the information of Order in the Delivery Form when customer confirms that
3	requestToPlaceRushOrder	void	Requesting Place Rush Order when user clicks the button “Place Rush Order”

			Receiving the entity Order
4	notifyInvalidDeliveryInfo	void	After checking the Delivery Info, notify to customer if the info is invald or left-blank
5	checkPlaceRushOrder	boolean	Checking whether customer has choose placing rush order
6	notifyNotPlaceRushOrder	void	Notify that customer didn’t want to place rush order, continue placing normal order
7	notifyNotSupportedMedia	void	Notify media is not supported for rush order, allowing the user to change Media
8	notifyNotSupportedAddress	void	Notify Address is not supported for shipping in rush order, allowing the user to change Address
9	submitDeliveryForm	void	Saving the Delivery Form for customer choosing to place rush order
10	checkSupportedAddress	boolean	Checking whether the Address is supported for shipping in rush order
11	notifyInvalidInput	void	Notify that the customer has entered invalid information

Parameter:

- None

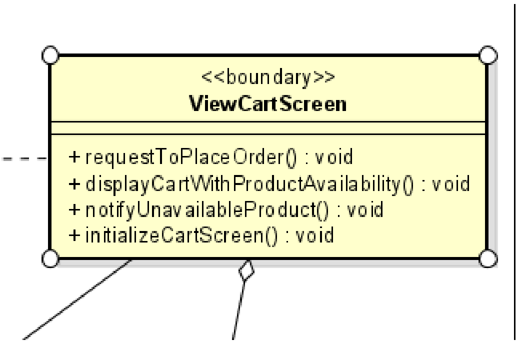
Exception:

- None

Method

- None

6. Class “ViewCartScreen”


<<boundary>>
ViewCartScreen
+ requestToPlaceOrder() : void
+ displayCartWithProductAvailability() : void
+ notifyUnavailableProduct() : void
+ initializeCartScreen() : void

Attribute

- None

Operation

#	Name	Return type	Description (purpose)
1	requestToPlaceOrder	Cart	Saving the information of Media when users choose Cart Screen after clicking the button “Place Order”
2	displayCartWithProductAvailability	void	Displaying the Media that exists in the Cart
3	initializeCartScreen	void	Initializing Cart Screen
4	notifyUnavailableProduct	void	Notify that media is unavailable to order

Parameter:

- None

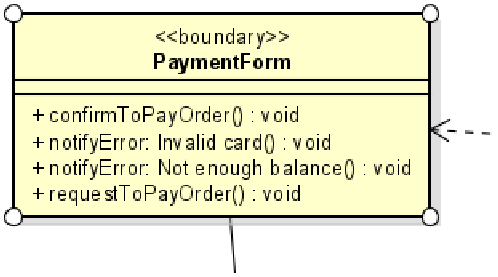
Exception:

- None

Method

- None

7. Class “PaymentForm”


<<boundary>>
PaymentForm
+ confirmToPayOrder() : void
+ notifyError: Invalid card() : void
+ notifyError: Not enough balance() : void
+ requestToPayOrder() : void

Attribute

- None

Operation

#	Name	Return type	Description (purpose)
1	requestToPayOrder	void	Taking the actions of paying Order, receiving the er Invoice
2	confirmToPayOrder	void	Saving the information of Invoice when users clicks button “Confirm Payment” Receiving the entity Invoice
3	notifyError: Invalid card()	void	Notifying the Error: Invalid card
4	notifyError: Not enough balance()	void	Notifying the Error: Not enough balance

Parameter:

- None

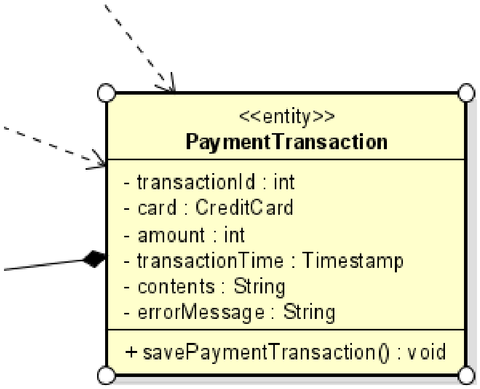
Exception:

- None

Method

- None

8. Class “PaymentTransaction”



Attribute

#	Name	Data type	Default value	Description
1	transactionId	int	NULL	Represent the Id transaction used for payment
2	card	CreditCard	NULL	Represent the information of Card used for payment
3	amount	int	NULL	Represent the amount of the payment
4	transactionTime	Timestamp	NULL	Represent the transaction time of the payment
5	content	String	NULL	Represent the content of the payment
6	errorMessage	String	NULL	Represent the error during the transaction

Operation

#	Name	Return type	Description (purpose)
1	savePaymentTransaction	void	Saving the payment transaction

Parameter:

- None

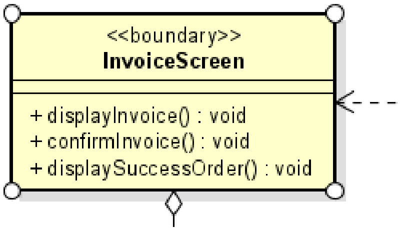
Exception:

- None

Method

- None

9. Class “InvoiceScreen”


<<boundary>>
InvoiceScreen
+ displayInvoice() : void
+ confirmInvoice() : void
+ displaySuccessOrder() : void

Attribute

- None

Operation

#	Name	Return type	Description (purpose)
1	displayInvoice	void	Displaying the InvoiceScreen and receiving the enti Invoice
2	confirmInvoice	void	Saving the information of Invoice when user clicks button “Confirm Invoice”
3	displaySuccessOrder	void	Displaying the notification of Success Order after u completes the actions of place order

Parameter:

- None

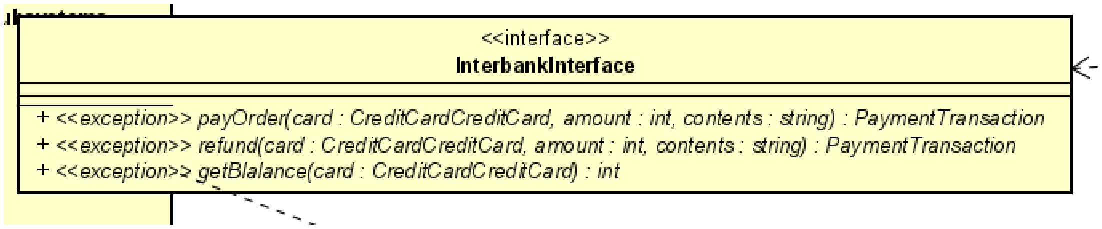
Exception:

- None

Method

- None

10. Class “InterbankInterface”


<<interface>>
InterbankInterface
+ <<exception>> payOrder(card : CreditCardCreditCard, amount : int, contents : string) : PaymentTransaction
+ <<exception>> refund(card : CreditCardCreditCard, amount : int, contents : string) : PaymentTransaction
+ <<exception>> getBlalance(card : CreditCardCreditCard) : int

Attribute

- None

Operation

#	Name	Return type	Description (purpose)
1	payOrder	PaymentTransaction	Pay order, and then return the payment transaction
2	refund	PaymentTransaction	Refund, and then return the payment transaction
3	getBalance	int	Return the balance amount in the CreditCard to customer

Parameter:

- card - the credit card used for payment/refund
- amount - the amount to pay/refund
- contents - the transaction contents

Exception:

- UnrecognizedException - if responded with an unknown error code or something goes wrong
- PaymentException - if responded with a pre-defined error code

Method

- None

State

- None