2.1. CREATE INITIAL DESIGN CLASSES

2.1.1. Design Boundary Classes

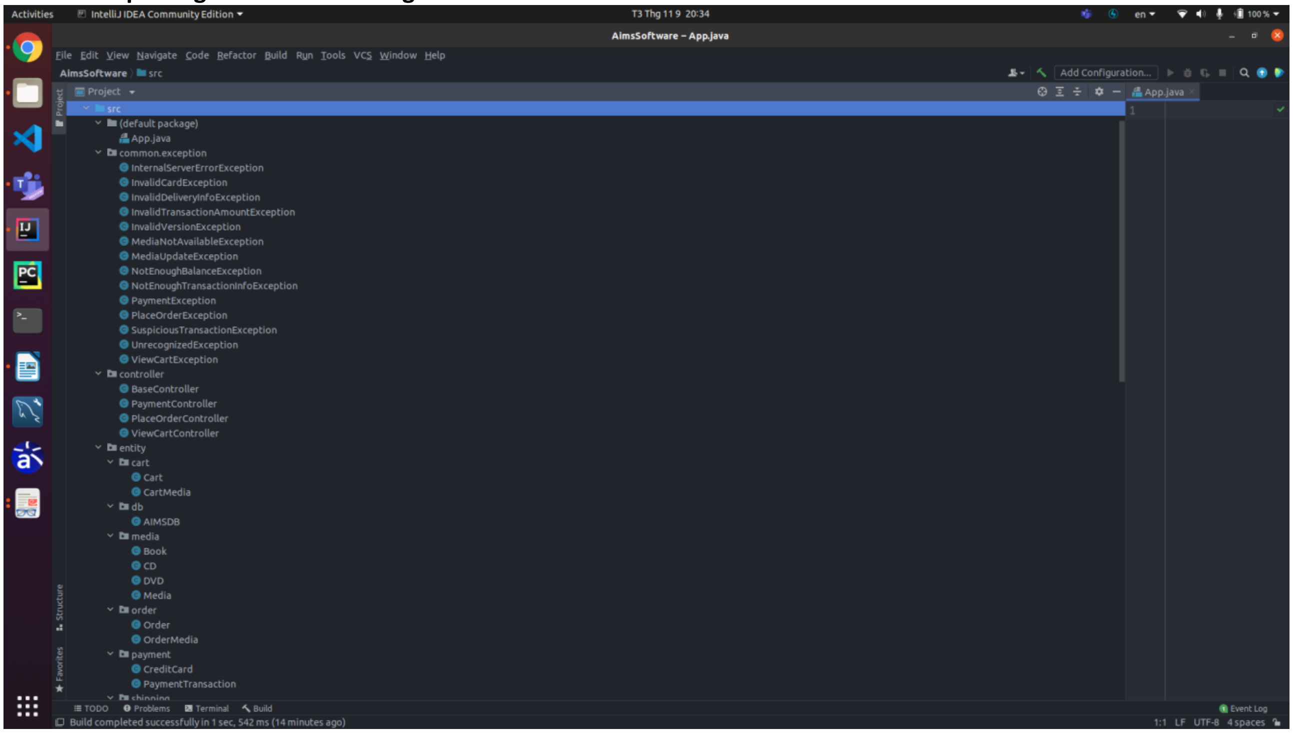
User interface (UI) boundary classes

System/device boundary classes

2.1.2. Design Entity Classes

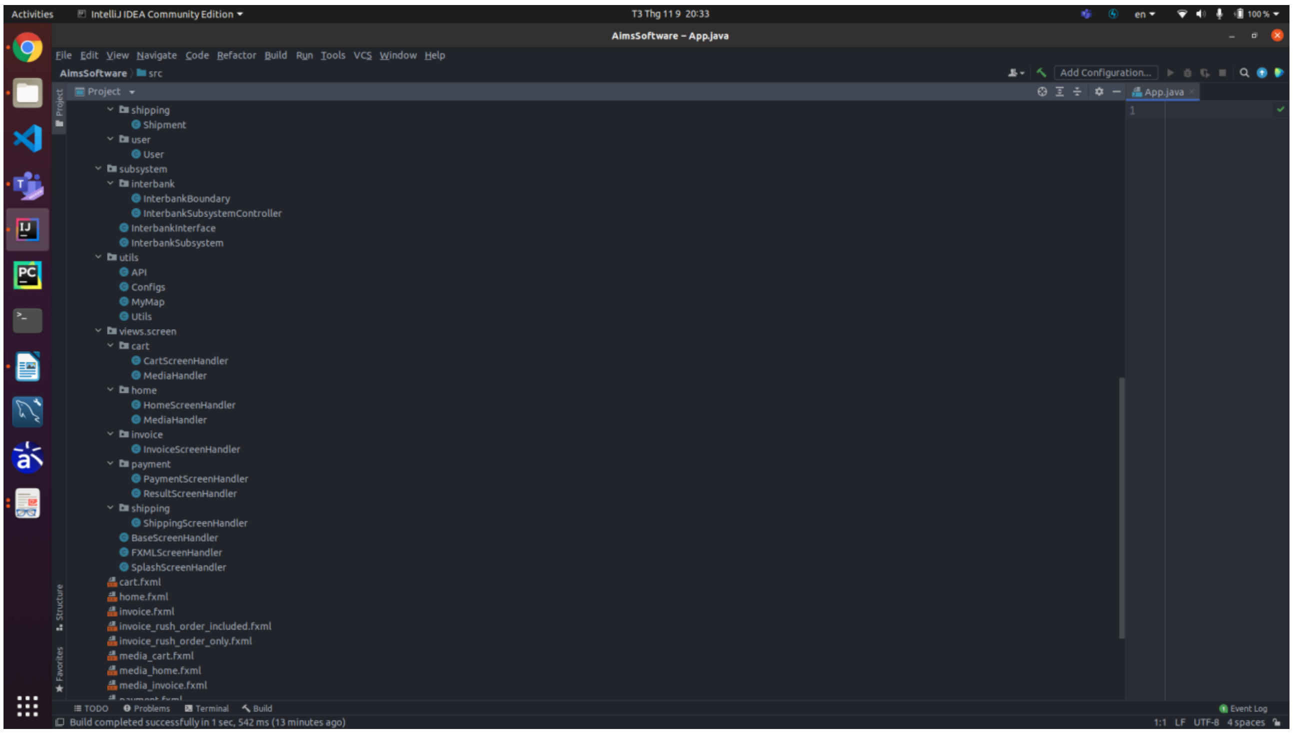
2.1.3. Design Control Classes

2.1.4. Group Design Classes in Packages



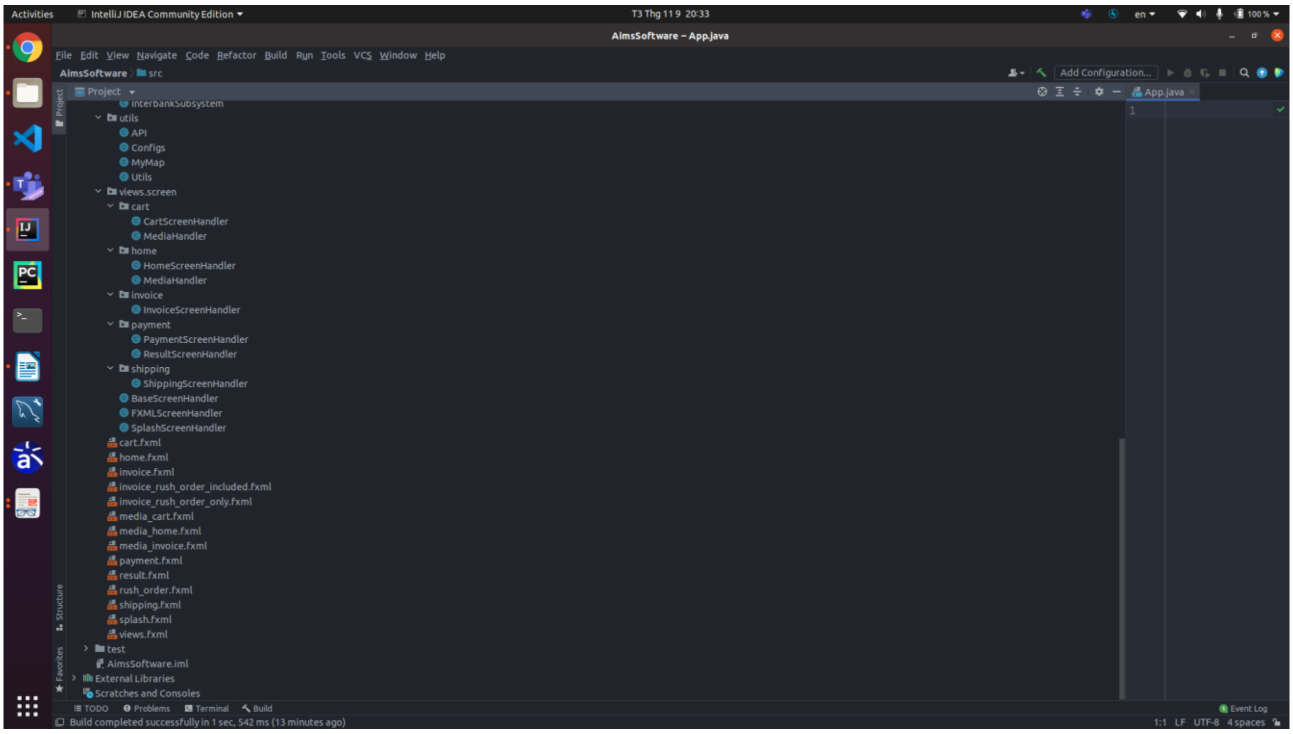
```
src
  (default package)
    App.java
  common.exception
    InternalServerErrorException
    InvalidCardException
    InvalidDeliveryInfoException
    InvalidTransactionAmountException
    InvalidVersionException
    MediaNotAvailableException
    MediaUpdateException
    NotEnoughBalanceException
    NotEnoughTransactionInfoException
    PaymentException
    PlaceOrderException
    SuspiciousTransactionException
    UnrecognizedException
    ViewCartException
  controller
    BaseController
    PaymentController
    PlaceOrderController
    ViewCartController
  entity
    cart
      Cart
      CartMedia
    db
      AIMSDB
    media
      Book
      CD
      DVD
      Media
    order
      Order
      OrderMedia
    payment
      CreditCard
      PaymentTransaction
    shipping
```

Build completed successfully in 1 sec, 542 ms (14 minutes ago)



```
    shipping
      Shipment
    user
      User
  subsystem
    interbank
      InterbankBoundary
      InterbankSubsystemController
    InterbankInterface
    InterbankSubsystem
  utils
    API
    Configs
    MyMap
    Utils
  views.screen
    cart
      CartScreenHandler
      MediaHandler
    home
      HomeScreenHandler
      MediaHandler
    invoice
      InvoiceScreenHandler
    payment
      PaymentScreenHandler
      ResultScreenHandler
    shipping
      ShippingScreenHandler
    BaseScreenHandler
    FXMLScreenHandler
    SplashScreenHandler
  cart.fxml
  home.fxml
  invoice.fxml
  invoice_rush_order_included.fxml
  invoice_rush_order_only.fxml
  media_cart.fxml
  media_home.fxml
  media_invoice.fxml
```

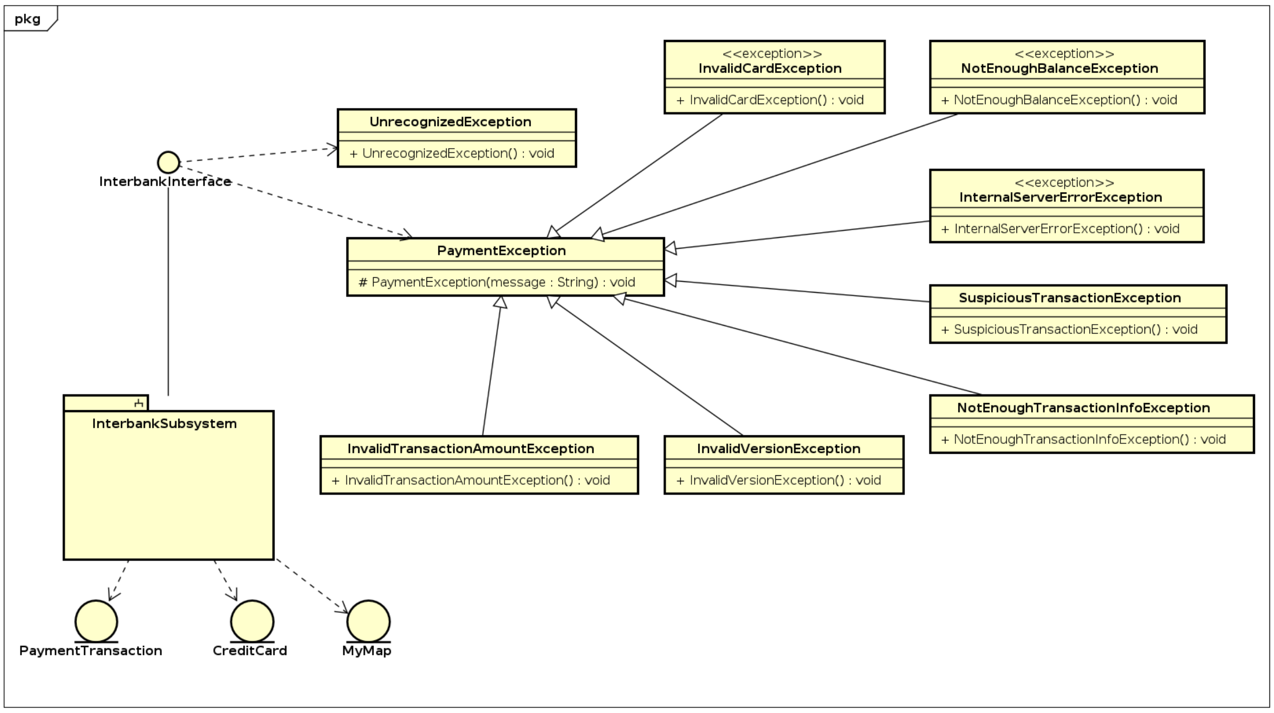
Build completed successfully in 1 sec, 542 ms (13 minutes ago)

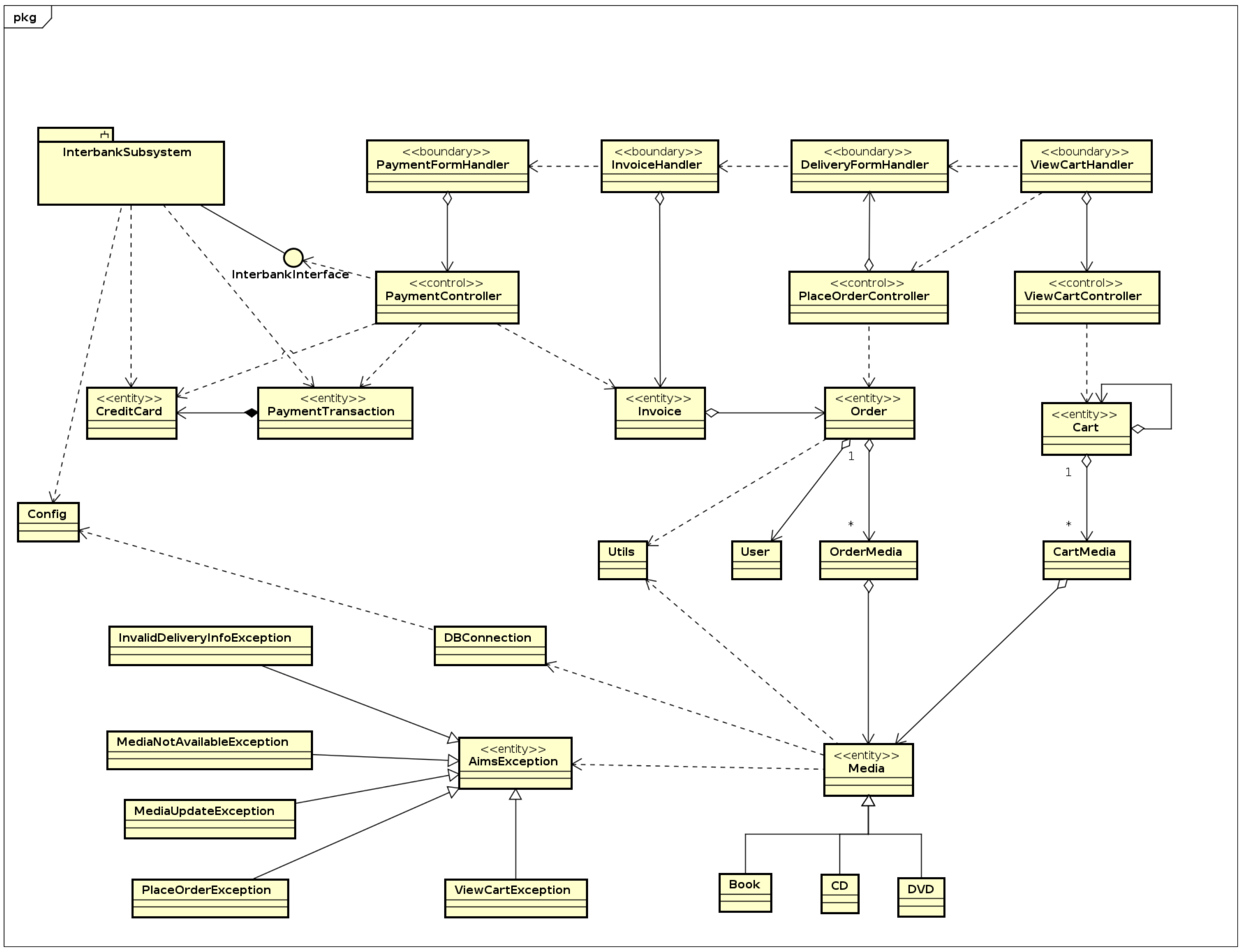


```
InterbankSubsystem
utils
    API
    Configs
    MyMap
    Utils
views.screen
    cart
        CartScreenHandler
        MediaHandler
    home
        HomeScreenHandler
        MediaHandler
    invoice
        InvoiceScreenHandler
    payment
        PaymentScreenHandler
        ResultScreenHandler
    shipping
        ShippingScreenHandler
    BaseScreenHandler
    FXMLScreenHandler
    SplashScreenHandler
cart.fxml
home.fxml
invoice.fxml
invoice_rush_order_included.fxml
invoice_rush_order_only.fxml
media_cart.fxml
media_home.fxml
media_invoice.fxml
payment.fxml
result.fxml
rush_order.fxml
shipping.fxml
splash.fxml
views.fxml
test
AimsSoftware.iml
External Libraries
Scratches and Consoles
```

Build completed successfully in 1 sec, 542 ms (13 minutes ago)

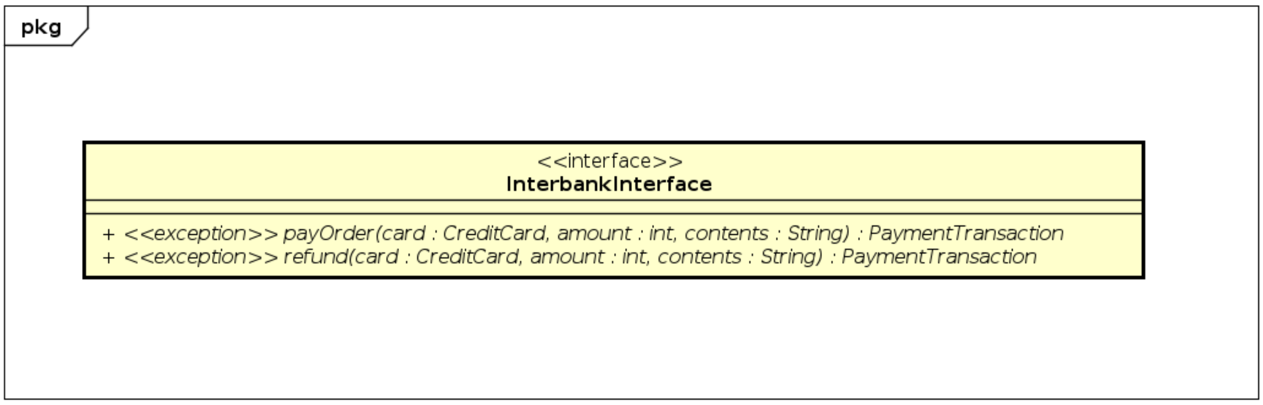
2.2. DEFINE RELATIONSHIPS BETWEEN CLASSES


pkg
<<exception>>
InvalidCardException
+ InvalidCardException() : void
<<exception>>
NotEnoughBalanceException
+ NotEnoughBalanceException() : void
UnrecognizedException
+ UnrecognizedException() : void
InterbankInterface
<<exception>>
InternalServerErrorException
+ InternalServerErrorException() : void
PaymentException
# PaymentException(message : String) : void
SuspiciousTransactionException
+ SuspiciousTransactionException() : void
NotEnoughTransactionInfoException
+ NotEnoughTransactionInfoException() : void
InterbankSubsystem
InvalidTransactionAmountException
+ InvalidTransactionAmountException() : void
InvalidVersionException
+ InvalidVersionException() : void
PaymentTransaction
CreditCard
MyMap


pkg
InterbankSubsystem
<<boundary>> PaymentFormHandler
<<boundary>> InvoiceHandler
<<boundary>> DeliveryFormHandler
<<boundary>> ViewCartHandler
InterbankInterface
<<control>> PaymentController
<<control>> PlaceOrderController
<<control>> ViewCartController
<<entity>> CreditCard
<<entity>> PaymentTransaction
<<entity>> Invoice
<<entity>> Order
<<entity>> Cart
Config
Utils
User
OrderMedia
CartMedia
1
*
1
*
InvalidDeliveryInfoException
DBConnection
MediaNotAvailableException
<<entity>> AimsException
<<entity>> Media
MediaUpdateException
PlaceOrderException
ViewCartException
Book
CD
DVD

2.3. CLASS DESIGN

2.3.1. Class “InterbankInterface”


pkg
<<interface>>
InterbankInterface
+ <<exception>> payOrder(card : CreditCard, amount : int, contents : String) : PaymentTransaction
+ <<exception>> refund(card : CreditCard, amount : int, contents : String) : PaymentTransaction

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

Parameter:

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

Exception:

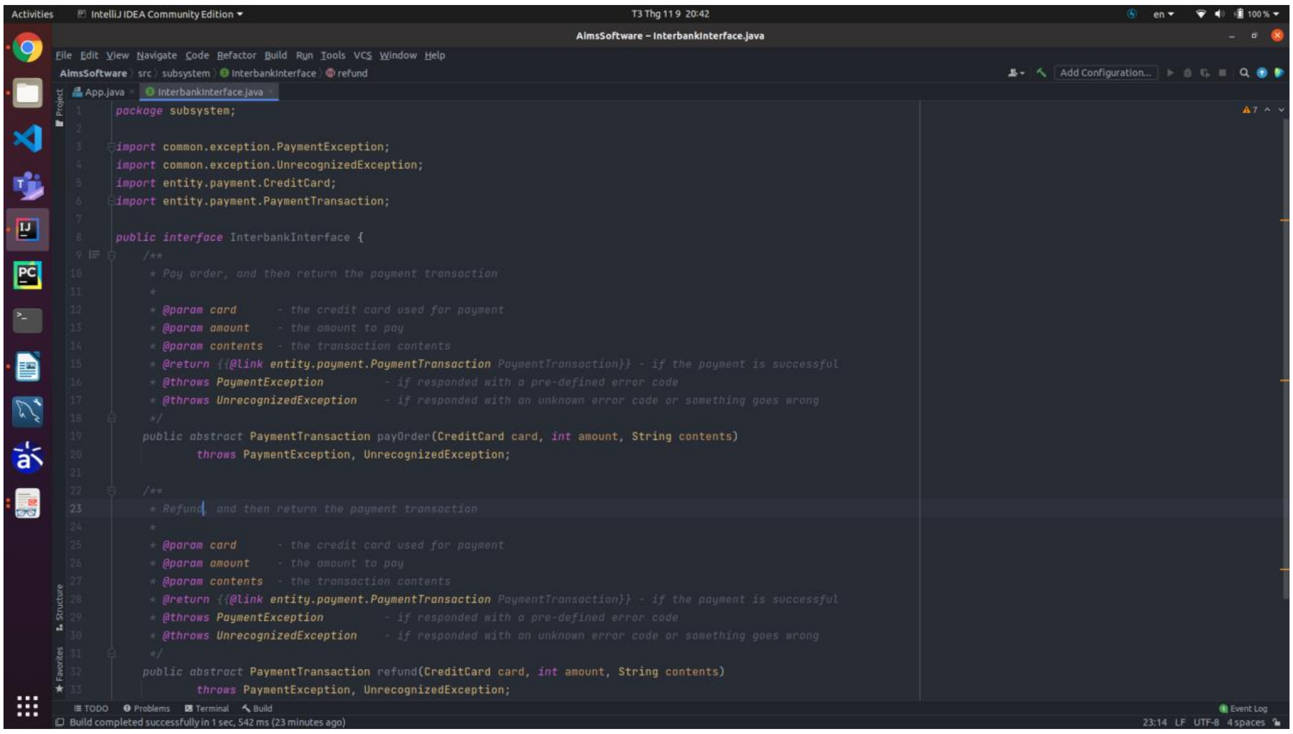
- PaymentException – if responded with a pre-defined error code
- UnrecognizedException – if responded with an unknown error code or something goes wrong

Method

None

State

None



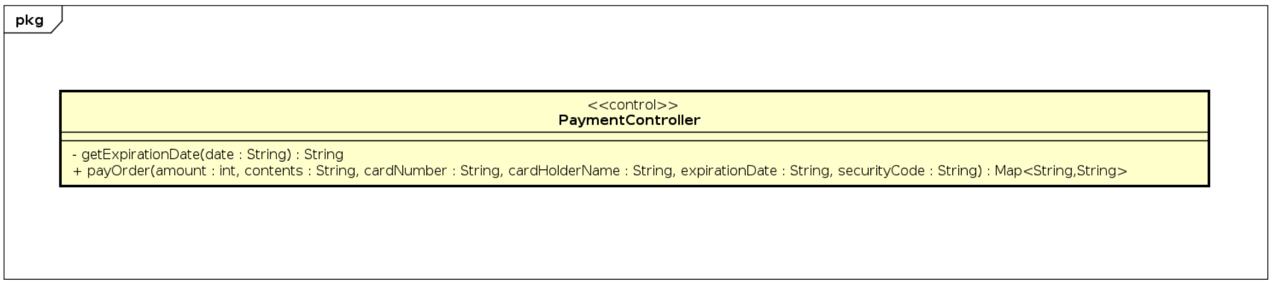
Activities
IntelliJ IDEA Community Edition
T3 Thg 11 9 20:42
AimsSoftware – InterbankInterface.java
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
AimsSoftware 〉 src 〉 subsystem 〉 InterbankInterface 〉 refund
Add Configuration...
App.java
InterbankInterface.java
TODO
Problems
Terminal
Build
Event Log

```java
package subsystem;

import common.exception.PaymentException;
import common.exception.UnrecognizedException;
import entity.payment.CreditCard;
import entity.payment.PaymentTransaction;

public interface InterbankInterface {
    /**
     * Pay order, and then return the payment transaction
     *
     * @param card      - the credit card used for payment
     * @param amount    - the amount to pay
     * @param contents  - the transaction contents
     * @return {{@link entity.payment.PaymentTransaction PaymentTransaction}} - if the payment is successful
     * @throws PaymentException         - if responded with a pre-defined error code
     * @throws UnrecognizedException    - if responded with an unknown error code or something goes wrong
     */
    public abstract PaymentTransaction payOrder(CreditCard card, int amount, String contents)
            throws PaymentException, UnrecognizedException;

    /**
     * Refund, and then return the payment transaction
     *
     * @param card      - the credit card used for payment
     * @param amount    - the amount to pay
     * @param contents  - the transaction contents
     * @return {{@link entity.payment.PaymentTransaction PaymentTransaction}} - if the payment is successful
     * @throws PaymentException         - if responded with a pre-defined error code
     * @throws UnrecognizedException    - if responded with an unknown error code or something goes wrong
     */
    public abstract PaymentTransaction refund(CreditCard card, int amount, String contents)
            throws PaymentException, UnrecognizedException;
```

Build completed successfully in 1 sec, 542 ms (23 minutes ago)

23:14 LF UTF-8 4 spaces

2.3.2. Class "PaymentController"



pkg
<<control>>
PaymentController
- getExpirationDate(date : String) : String
+ payOrder(amount : int, contents : String, cardNumber : String, cardHolderName : String, expirationDate : String, securityCode : String) : Map<String,String>

Attribute

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

Operation

Parameter:

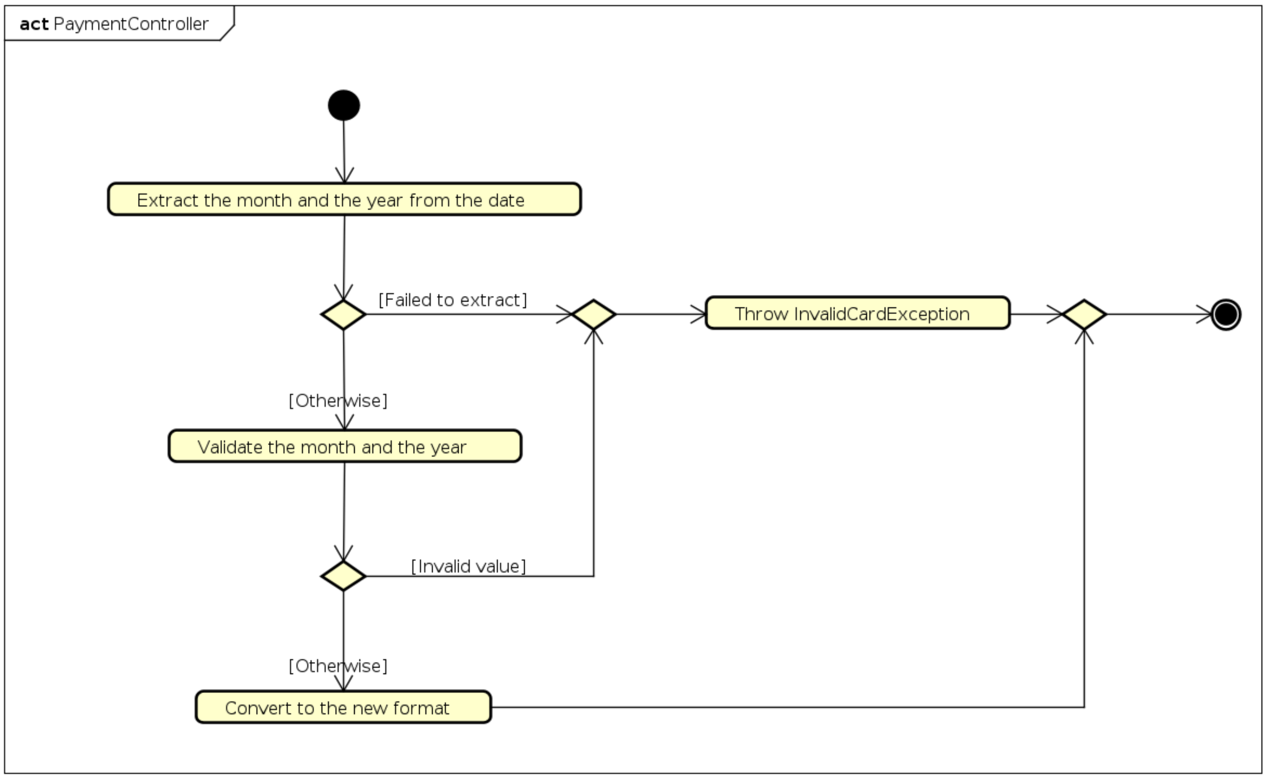
- amount – the amount to pay
- contents – the transaction contents
- cardNumber – the card number
- cardHolderName – the card holder name
- expirationDate – the expiration date in the format "mm/yy"
- securityCode – the cvv/cvc code of the credit card

Exception:

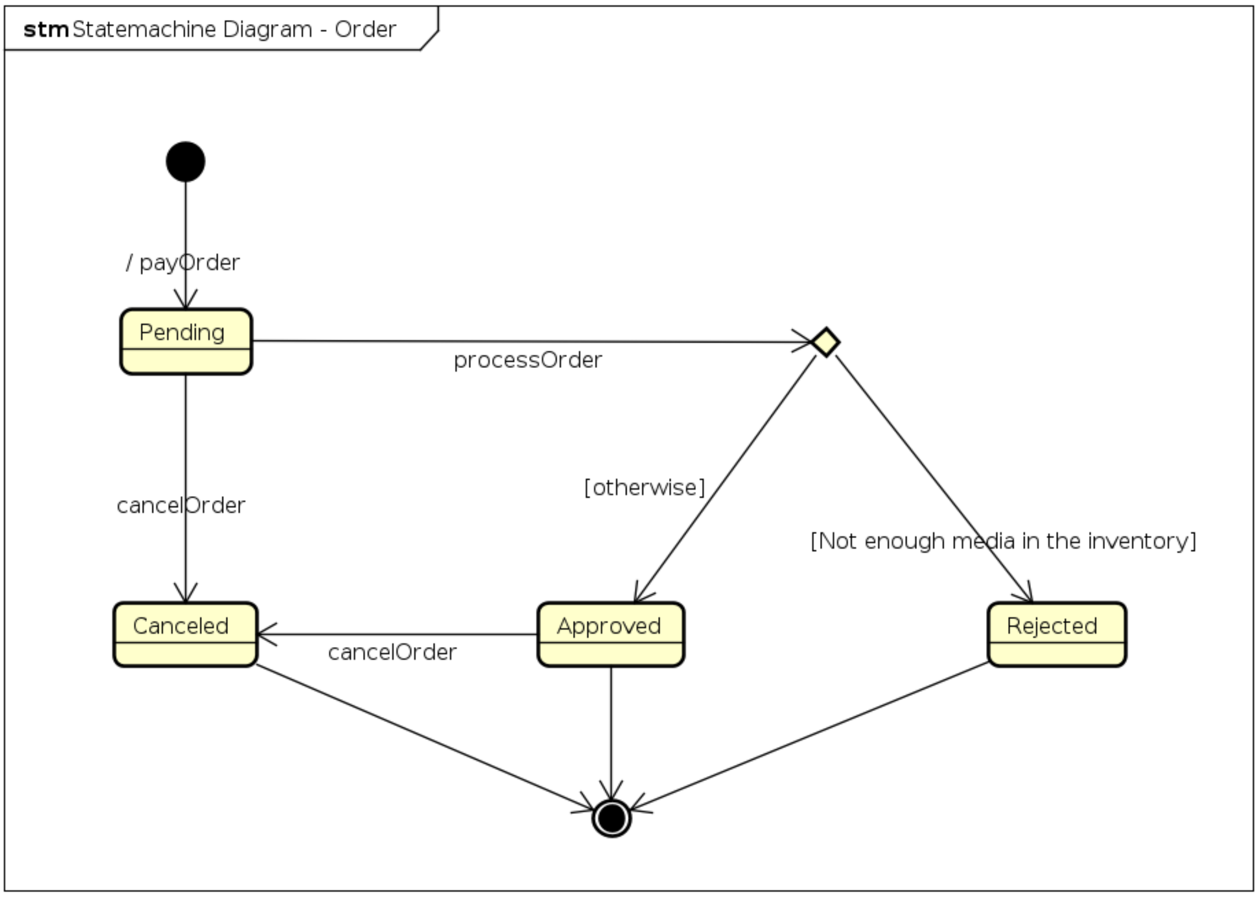
None

Method

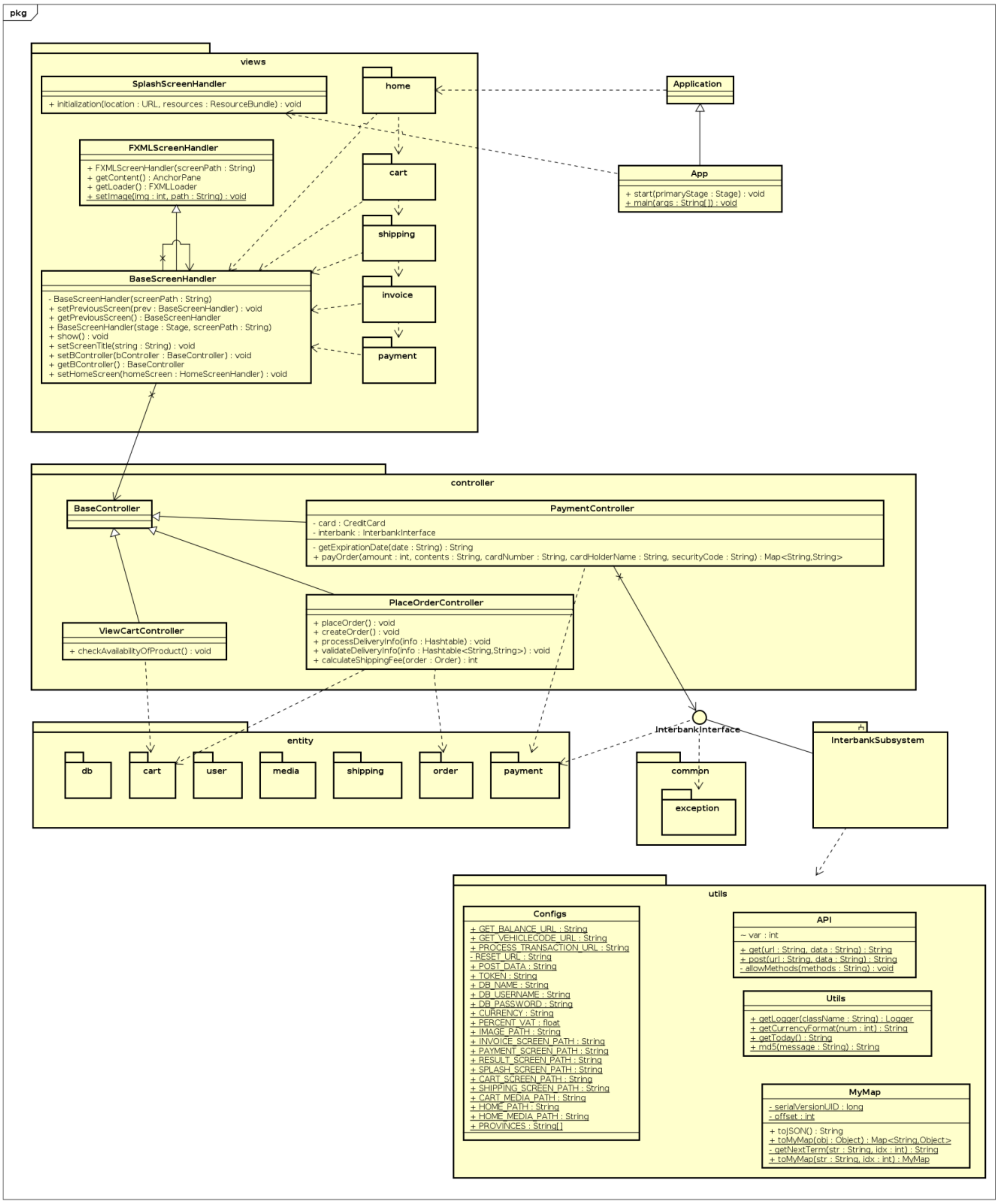
- getExpirationDate: Given the String "date" representing the expiration date in the format "mm/yy", this method convert it into the required format "mmyy". The algorithm is illustrated as follows.


act PaymentController
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

2.3.3. State Machine Diagram for an "Order" object

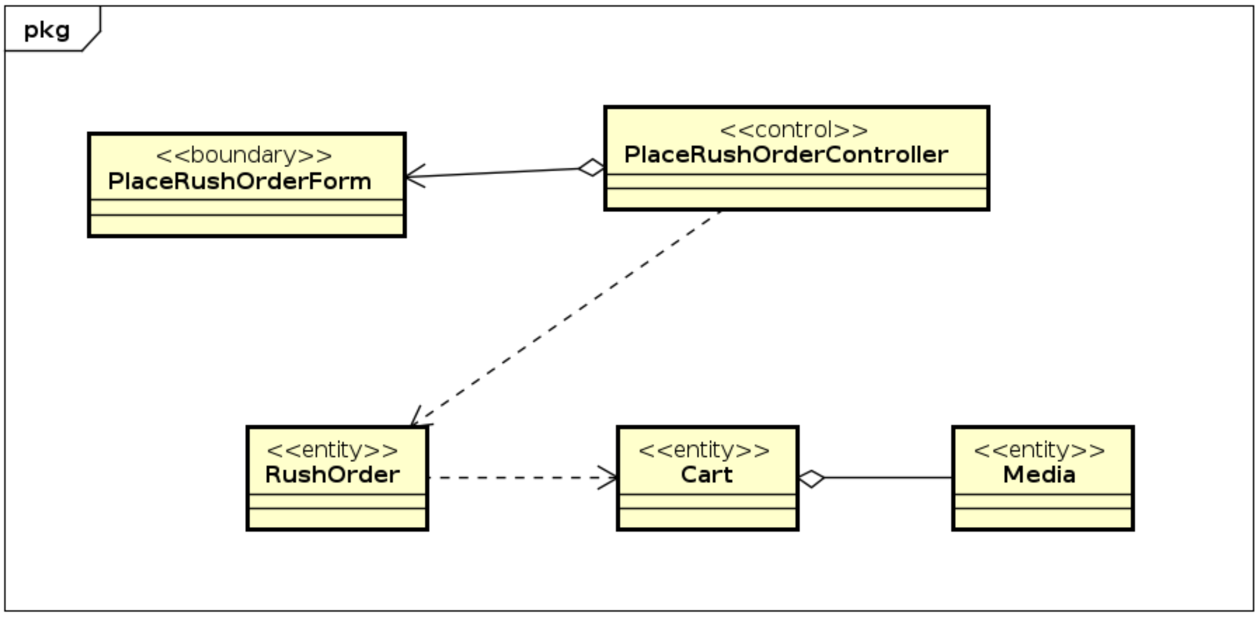


2.4. CLASS DIAGRAM

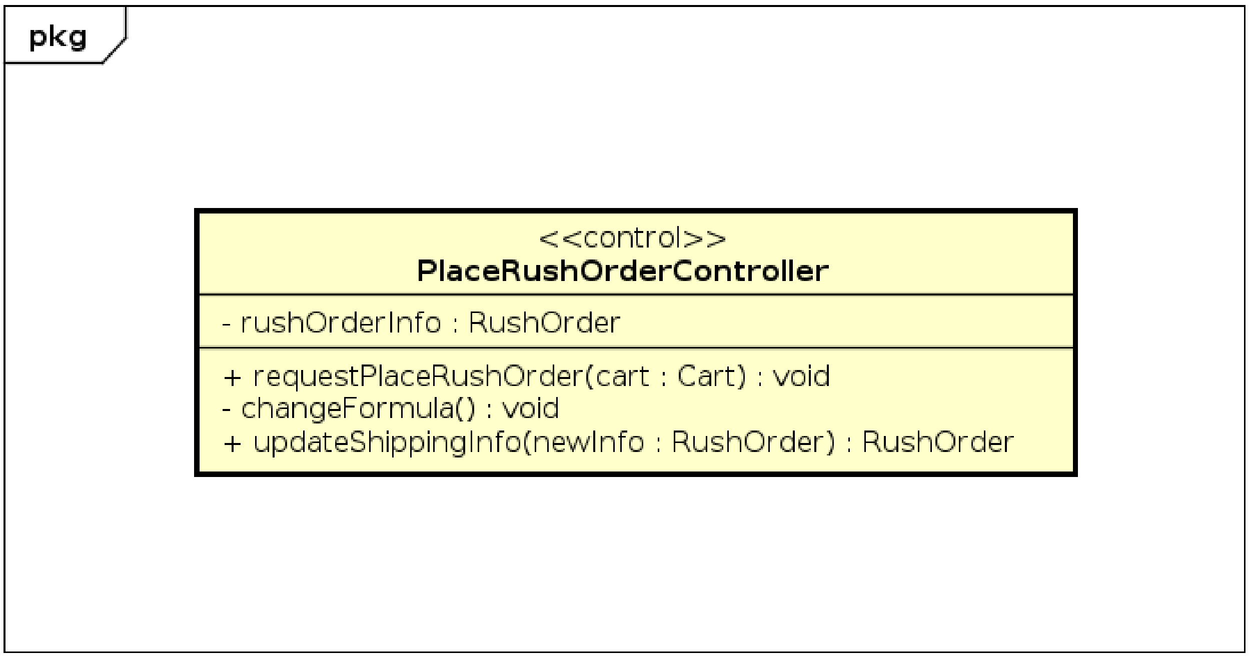

pkg
views
SplashScreenHandler
+ initialization(location : URL, resources : ResourceBundle) : void
home
cart
shipping
invoice
payment
Application
App
+ start(primaryStage : Stage) : void
+ main(args : String[]) : void
FXMLScreenHandler
+ FXMLScreenHandler(screenPath : String)
+ getContent() : AnchorPane
+ getLoader() : FXMLLoader
+ setImage(img : int, path : String) : void
BaseScreenHandler
- BaseScreenHandler(screenPath : String)
+ setPreviousScreen(prev : BaseScreenHandler) : void
+ getPreviousScreen() : BaseScreenHandler
+ BaseScreenHandler(stage : Stage, screenPath : String)
+ show() : void
+ setScreenTitle(string : String) : void
+ setBController(bController : BaseController) : void
+ getBController() : BaseController
+ setHomeScreen(homeScreen : HomeScreenHandler) : void
controller
BaseController
PaymentController
- card : CreditCard
- interbank : InterbankInterface
- getExpirationDate(date : String) : String
+ payOrder(amount : int, contents : String, cardNumber : String, cardHolderName : String, securityCode : String) : Map<String,String>
PlaceOrderController
+ placeOrder() : void
+ createOrder() : void
+ processDeliveryInfo(info : Hashtable) : void
+ validateDeliveryInfo(info : Hashtable<String,String>) : void
+ calculateShippingFee(order : Order) : int
ViewCartController
+ checkAvailabilityOfProduct() : void
entity
db
cart
user
media
shipping
order
payment
InterbankInterface
common
exception
InterbankSubsystem
utils
Configs
+ GET_BALANCE_URL : String
+ GET_VEHICLECODE_URL : String
+ PROCESS_TRANSACTION_URL : String
- RESET_URL : String
+ POST_DATA : String
+ TOKEN : String
+ DB_NAME : String
+ DB_USERNAME : String
+ DB_PASSWORD : String
+ CURRENCY : String
+ PERCENT_VAT : float
+ IMAGE_PATH : String
+ INVOICE_SCREEN_PATH : String
+ PAYMENT_SCREEN_PATH : String
+ RESULT_SCREEN_PATH : String
+ SPLASH_SCREEN_PATH : String
+ CART_SCREEN_PATH : String
+ SHIPPING_SCREEN_PATH : String
+ CART_MEDIA_PATH : String
+ HOME_PATH : String
+ HOME_MEDIA_PATH : String
+ PROVINCES : String[]
API
~ var : int
+ get(url : String, data : String) : String
+ post(url : String, data : String) : String
- allowMethods(methods : String) : void
Utils
+ getLogger(className : String) : Logger
+ getCurrencyFormat(num : int) : String
+ getToday() : String
+ md5(message : String) : String
MyMap
- serialVersionUID : long
- offset : int
+ toJSON() : String
+ toMyMap(obj : Object) : Map<String,Object>
- getNextTerm(str : String, idx : int) : String
+ toMyMap(str : String, idx : int) : MyMap

2.5. CLASS DESIGN FOR "PLACE RUSH ORDER"

2.5.1. Define relationships between classes



2.5.2. Class design


pkg
<<control>>
PlaceRushOrderController
- rushOrderInfo : RushOrder
+ requestPlaceRushOrder(cart : Cart) : void
- changeFormula() : void
+ updateShippingInfo(newInfo : RushOrder) : RushOrder

Attribute

#	*Name*	*Data type*	*Default value*	*Description*
1	rushOrderInfo	RushOrder	NULL	Stores all the medias information and shipping infos of each media

Operation

#	Name	Return type	Description (purpose)
1	RequestPlaceRushOrder	void	Receive request to Place rush order if user chooses to place rush order
2	updateShippingInfo	RushOrder	Update shipping info based on new information from users

Parameter:

- cart: the cart of user
- newInfo: updated info from user

Exception:

- None

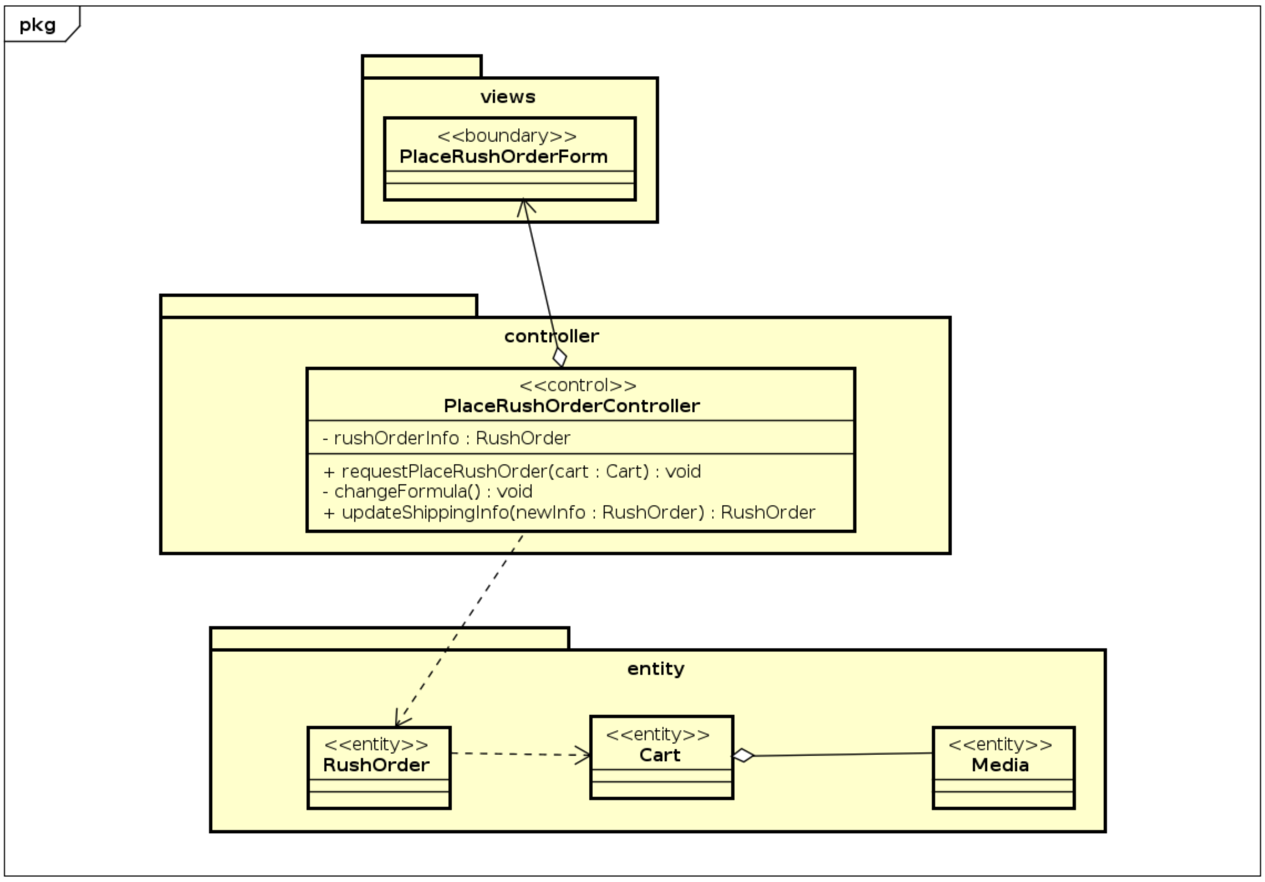
Method

- changeFormula: Because this is place rush order so the software must change the formula to calculate new shipping fee

State

None

2.5.3. Class diagram


pkg
views
<<boundary>>
PlaceRushOrderForm
controller
<<control>>
PlaceRushOrderController
- rushOrderInfo : RushOrder
+ requestPlaceRushOrder(cart : Cart) : void
- changeFormula() : void
+ updateShippingInfo(newInfo : RushOrder) : RushOrder
entity
<<entity>>
RushOrder
<<entity>>
Cart
<<entity>>
Media