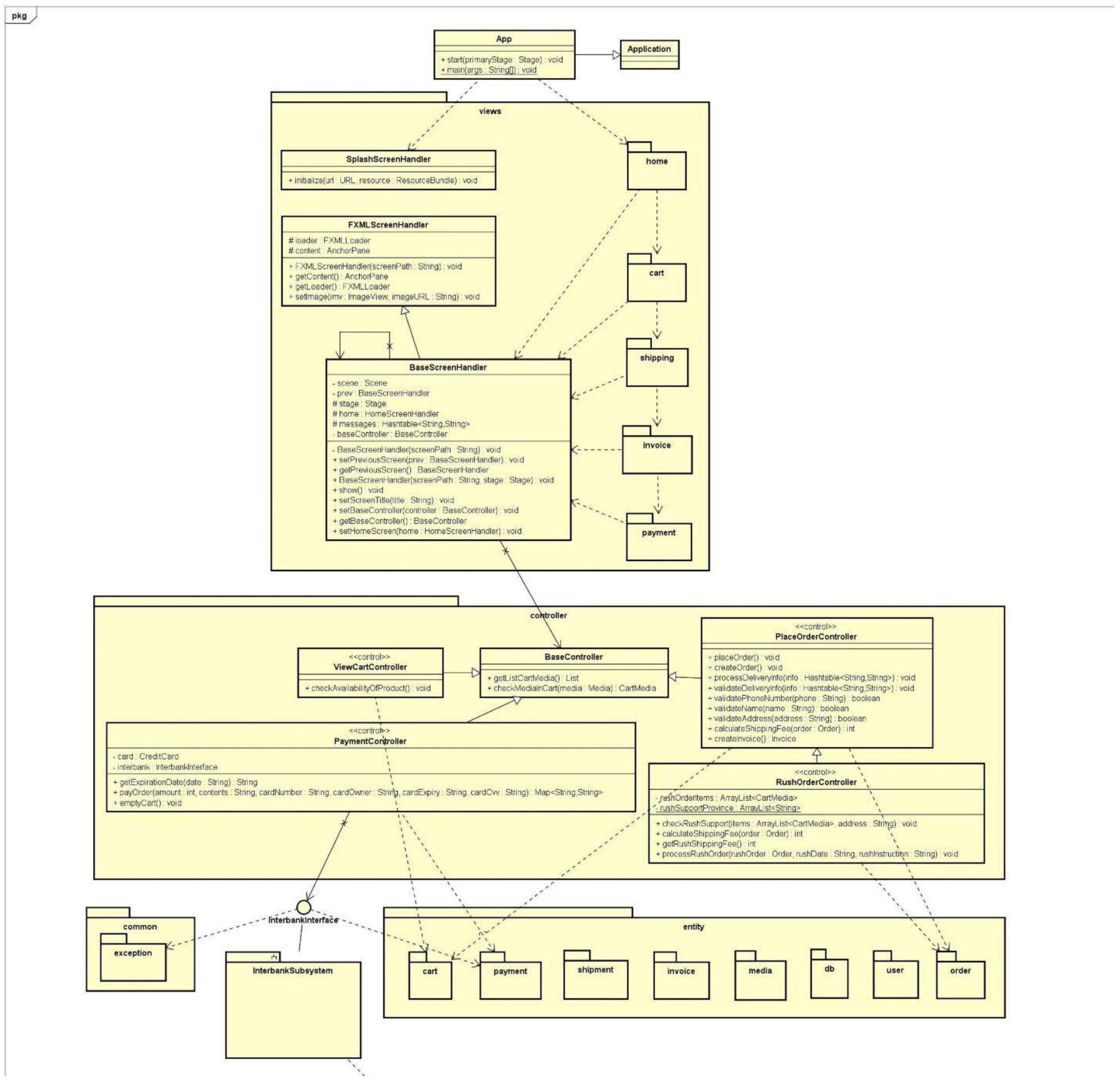
Lab 10 report

1. Coupling



pkg
App
+ start(primaryStage : Stage) : void
+ main(args : String[]) : void
Application
views
SplashScreenHandler
+ initialize(url : URL, resource : ResourceBundle) : void
FXMLScreenHandler
# loader : FXMLLoader
# content : AnchorPane
+ FXMLScreenHandler(screenPath : String) : void
+ getContent() : AnchorPane
+ getLoader() : FXMLLoader
+ setImage(imv : ImageView, imageURL : String) : void
BaseScreenHandler
- scene : Scene
- prev : BaseScreenHandler
# stage : Stage
# home : HomeScreenHandler
# messages : Hashtable<String,String>
- baseController : BaseController
- BaseScreenHandler(screenPath : String) : void
+ setPreviousScreen(prev : BaseScreenHandler) : void
+ getPreviousScreen() : BaseScreenHandler
+ BaseScreenHandler(screenPath : String, stage : Stage) : void
+ show() : void
+ setScreenTitle(title : String) : void
+ setBaseController(controller : BaseController) : void
+ getBaseController() : BaseController
+ setHomeScreen(home : HomeScreenHandler) : void
home
cart
shipping
invoice
payment
controller
<<control>>
ViewCartController
+ checkAvailabilityOfProduct() : void
BaseController
+ getListCartMedia() : List
+ checkMediaInCart(media : Media) : CartMedia
<<control>>
PlaceOrderController
+ placeOrder() : void
+ createOrder() : void
+ processDeliveryInfo(info : Hashtable<String,String>) : void
+ validateDeliveryInfo(info : Hashtable<String,String>) : void
+ validatePhoneNumber(phone : String) : boolean
+ validateName(name : String) : boolean
+ validateAddress(address : String) : boolean
+ calculateShippingFee(order : Order) : int
+ createInvoice() : Invoice
<<control>>
PaymentController
- card : CreditCard
- interbank : InterbankInterface
+ getExpirationDate(date : String) : String
+ payOrder(amount : int, contents : String, cardNumber : String, cardOwner : String, cardExpiry : String, cardCvv : String) : Map<String,String>
+ emptyCart() : void
<<control>>
RushOrderController
- rushOrderItems : ArrayList<CartMedia>
- rushSupportProvince : ArrayList<String>
+ checkRushSupport(items : ArrayList<CartMedia>, address : String) : void
+ calculateShippingFee(order : Order) : int
+ getRushShippingFee() : int
+ processRushOrder(rushOrder : Order, rushDate : String, rushInstruction : String) : void
common
exception
InterbankInterface
InterbankSubsystem
entity
cart
payment
shipment
invoice
media
db
user
order

The MVC design pattern is used in AIMS project. This means that the project is organized into layers, or controllers, that handle different parts of the work:

- The classes in this program belong to different main modules: views, controller, and entity, which leads to being apparent in the tasks of classes .This will make it easier to keep your code organized and minimize the number of dependencies between the classes.

- The communication between the modules is done through base classes (BaseController and BaseScreenHandler). This makes it easier for the modules to communicate with each other because they don't have to call each other directly.

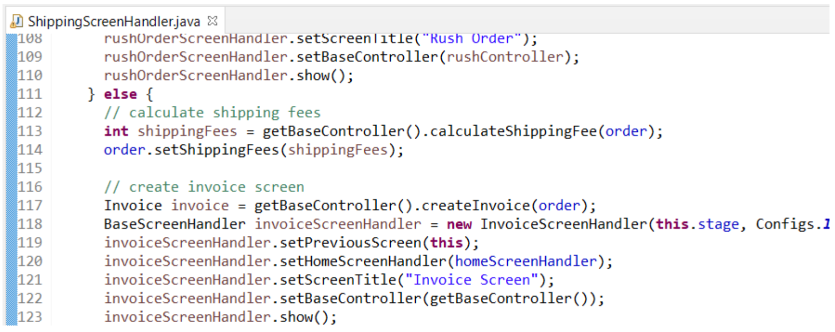
Preventing high because there is less connection among classes.

So, this is data coupling.

2. Cohesion

   Some problem about cohesions:

- ScreenHandler is responsible for handling requests from the user interface and sending method calls to the controller for processing. However, ScreenHandler is acting as the controller.
- ShippingScreenHandler helps you create an Order, while the PlaceOrderController should be responsible for setting up the Order.


ShippingScreenHandler.java

```java
            rushOrderScreenHandler.setScreenTitle("Rush Order");
            rushOrderScreenHandler.setBaseController(rushController);
            rushOrderScreenHandler.show();
        } else {
            // calculate shipping fees
            int shippingFees = getBaseController().calculateShippingFee(order);
            order.setShippingFees(shippingFees);

            // create invoice screen
            Invoice invoice = getBaseController().createInvoice(order);
            BaseScreenHandler invoiceScreenHandler = new InvoiceScreenHandler(this.stage, Configs.I
            invoiceScreenHandler.setPreviousScreen(this);
            invoiceScreenHandler.setHomeScreenHandler(homeScreenHandler);
            invoiceScreenHandler.setScreenTitle("Invoice Screen");
            invoiceScreenHandler.setBaseController(getBaseController());
            invoiceScreenHandler.show();
```

- PlaceOrderController:
  - There is a createInvoice() method but it is not related to any other methods in the class.
  - The PlaceOrder() method is not fully allocated tasks. Tasks are responsible for ScreenHandler.
- PaymentController: The getExpirationDate(String), emptyCart() method has no data or logic binding.
- InterbankSubsystem: The methods refund() and payOrder() have the same input parameters and return type. They can be used to refund or pay for an order separately.
- PlaceRushOrderController: The getRushTableMedia method is unrelated to other functions, which can be separated into util methods.

So, cohesion is not high