Sell_Fort

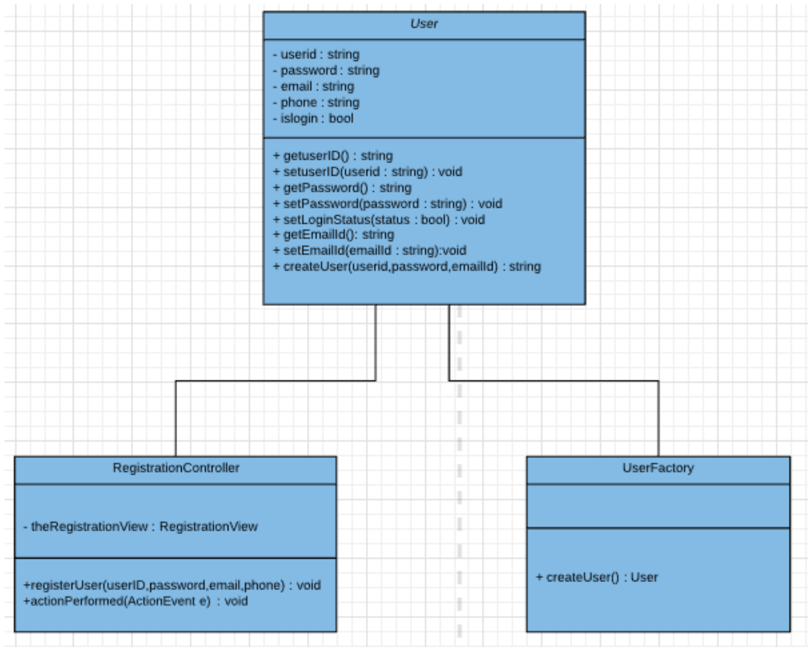
-Sell with Comfort

Team Members:

1. **Avinash Ratnavel Maharaj**
2. **Monika Tak**
3. **Srivishnu Alvakonda**

Following are the design patterns we implemented:

1) **Factory Method Design Pattern:**
   In our design, we have three types of users: Buyer, Seller and Admin. We introduced a UserFactory class which controls the instantiation of these users and refers to the newly created object using a common interface. For instance, in the future if we wish to extend the type of users and want to introduce a primeSeller/primeBuyer class, then the code modification will be easier because the object creation is being handled by one centralized class i.e. the UserFactory class.

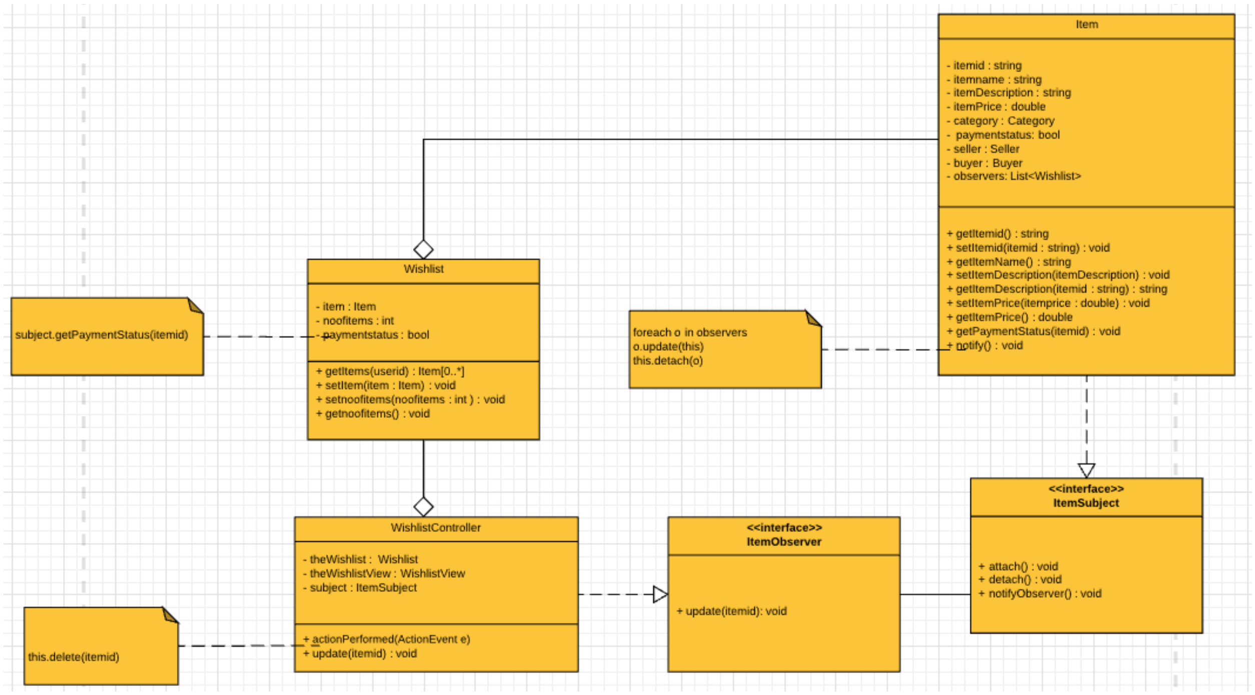

User
- userid : string
- password : string
- email : string
- phone : string
- islogin : bool
+ getuserID() : string
+ setuserID(userid : string) : void
+ getPassword() : string
+ setPassword(password : string) : void
+ setLoginStatus(status : bool) : void
+ getEmailId(): string
+ setEmailId(emailId : string):void
+ createUser(userid,password,emailId) : string
RegistrationController
- theRegistrationView : RegistrationView
+registerUser(userID,password,email,phone) : void
+actionPerformed(ActionEvent e) : void
UserFactory
+ createUser() : User

2) **Singleton Design Pattern:**
   We will be using the singleton design pattern for our Admin class incorporated in the UserFactory class itself. This is to make sure that there is only one instance of Admin class since there can only be one Admin for our application at a time.

3) **Observer Design Pattern:**

Requirement: While implementing, we came across the hurdle that when an item is sold to a buyer, at that time all the other buyers who have that same item in their wishList should be notified that the item has been already sold. The item should also be deleted from their wishList at the same time. For this reason, we adopted the Observer Design pattern wherein the wishListController class will act as an observer to the item class. Depending on the payment status of the item, the wishList object which has an array of Item objects should get modified for every buyer which has that particular item (which just got sold).

ItemSubject ≪Interface≫
ItemObserver ≪ Interface≫
Concrete Observer: WishListController
Concrete Subject: Item

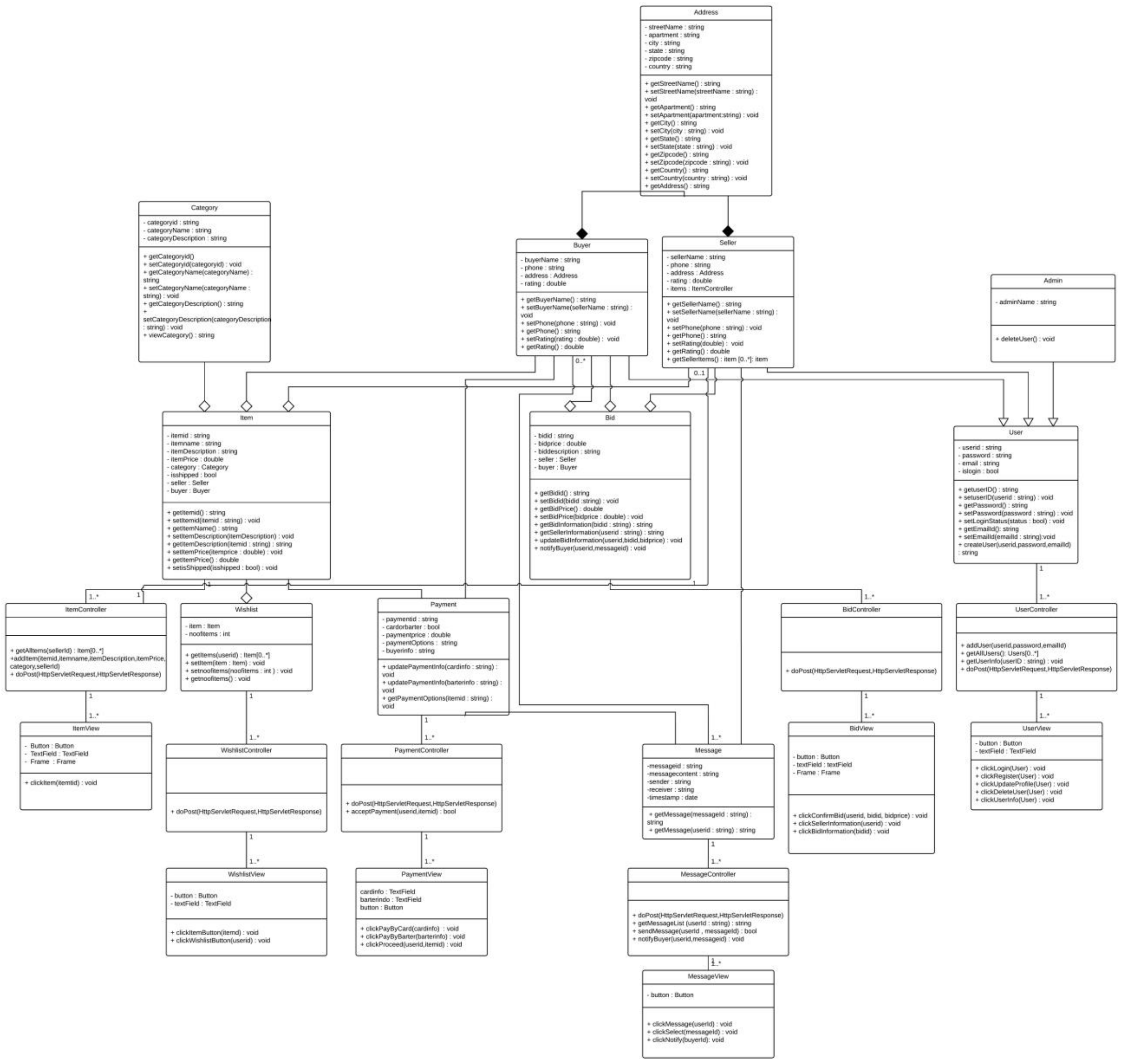


Refactoring:

1) Made User class as an abstract class since the various user types will have some common functions, so to avoid code duplication and to facilitate polymorphism we made the User class as abstract.
2) Added a few more class : LoginView , LoginController , RegistrationView , RegistrationController to suffice the requirements.
3) Changed the relationship between view and controller from association to aggregation which we observed while implementing the code. Since we are creating an object of the View class in our controller class.
4) Added few more attributes and methods to support all the functionalities in a few classes.

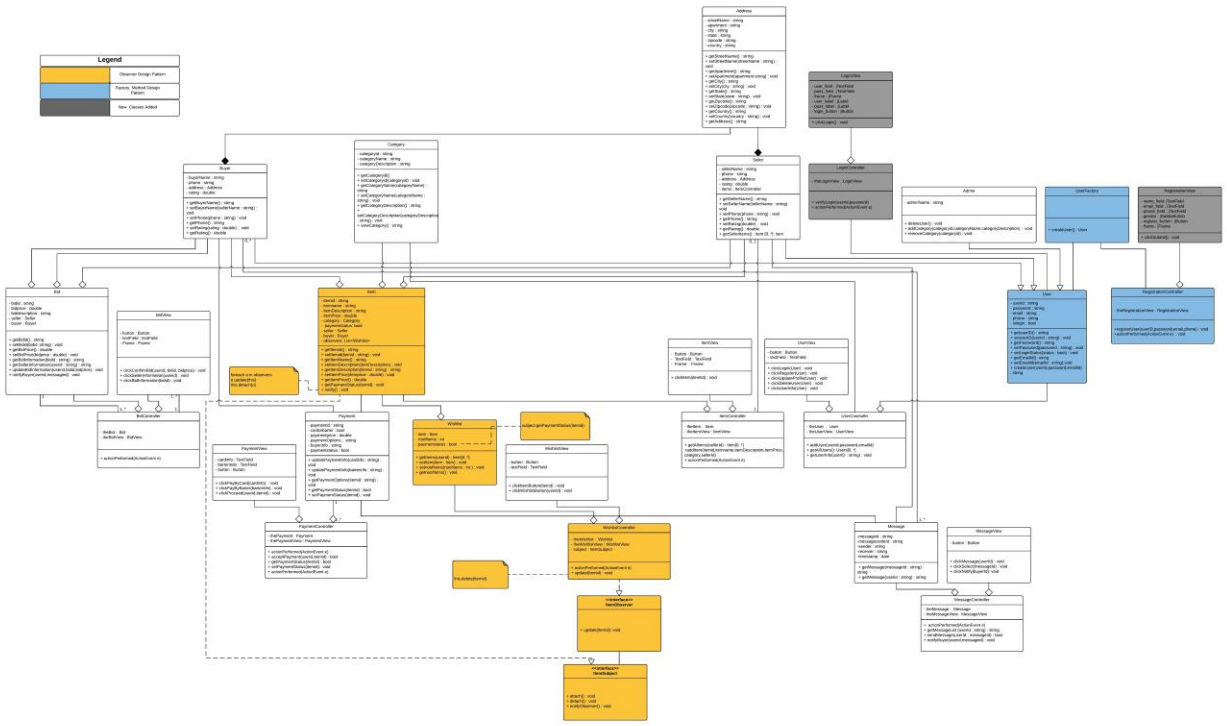
5) Restricted the ability to add and remove product categories to the User admin. For this we added the methods for adding/removing categories only in the admin class (inheriting from the User abstract class)

Old Class Diagram:
https://github.com/avra0601/CSCI-5448-OOAD-project/blob/master/Class%20Diagram.jpeg


Address
- streetName : string
- apartment : string
- city : string
- state : string
- zipcode : string
- country : string
+ getStreetName() : string
+ setStreetName(streetName : string) : void
+ getApartment() : string
+ setApartment(apartment:string) : void
+ getCity() : string
+ setCity(city : string) : void
+ getState() : string
+ setState(state : string) : void
+ getZipcode() : string
+ setZipcode(zipcode : string) : void
+ getCountry() : string
+ setCountry(country : string) : void
+ getAddress() : string
Category
- categoryid : string
- categoryName : string
- categoryDescription : string
+ getCategoryid()
+ setCategoryid(categoryid) : void
+ getCategoryName(categoryName) : string
+ setCategoryName(categoryName : string) : void
+ getCategoryDescription() : string
+ setCategoryDescription(categoryDescription : string) : void
+ viewCategory() : string
Buyer
- buyerName : string
- phone : string
- address : Address
- rating : double
+ getBuyerName() : string
+ setBuyerName(sellerName : string) : void
+ setPhone(phone : string) : void
+ getPhone() : string
+ setRating(rating : double) : void
+ getRating() : double
Seller
- sellerName : string
- phone : string
- address : Address
- rating : double
- items : ItemController
+ getSellerName() : string
+ setSellerName(sellerName : string) : void
+ setPhone(phone : string) : void
+ getPhone() : string
+ setRating(double) : void
+ getRating() : double
+ getSellerItems() : item [0..*]: item
Admin
- adminName : string
+ deleteUser() : void
0..*
0..1
Item
- itemid : string
- itemname : string
- itemDescription : string
- itemPrice : double
- category : Category
- isshipped : bool
- seller : Seller
- buyer : Buyer
+ getItemid() : string
+ setItemid(itemid : string) : void
+ getItemName() : string
+ setItemDescription(itemDescription) : void
+ getItemDescription(itemid : string) : string
+ setItemPrice(itemprice : double) : void
+ getItemPrice() : double
+ setisShipped(isshipped : bool) : void
Bid
- bidid : string
- bidprice : double
- biddescription : string
- seller : Seller
- buyer : Buyer
+ getBidid() : string
+ setBidid(bidid :string) : void
+ getBidPrice() : double
+ setBidPrice(bidprice : double) : void
+ getBidInformation(bidid : string) : string
+ getSellerInformation(userid : string) : string
+ updateBidInformation(userid,bidid,bidprice) : void
+ notifyBuyer(userid,messageid) : void
User
- userid : string
- password : string
- email : string
- islogin : bool
+ getuserID() : string
+ setuserID(userid : string) : void
+ getPassword() : string
+ setPassword(password : string) : void
+ setLoginStatus(status : bool) : void
+ getEmailId(): string
+ setEmailId(emailId : string):void
+ createUser(userid,password,emailId) : string
1
1..*
1
1
1..*
ItemController
+ getAllItems(sellerId) : Item[0..*]
+addItem(itemid,itemname,itemDescription,itemPrice, category,sellerId)
+ doPost(HttpServletRequest,HttpServletResponse)
Wishlist
- item : Item
- noofitems : int
+ getItems(userid) : Item[0..*]
+ setItem(item : Item) : void
+ setnoofitems(noofitems : int) : void
+ getnoofitems() : void
Payment
- paymentid : string
- cardorbarter : bool
- paymentprice : double
- paymentOptions : string
- buyerinfo : string
+ updatePaymentInfo(cardinfo : string) : void
+ updatePaymentInfo(barterinfo : string) : void
+ getPaymentOptions(itemid : string) : void
BidController
+ doPost(HttpServletRequest,HttpServletResponse)
UserController
+ addUser(userid,password,emailId)
+ getAllUsers(): Users[0..*]
+ getUserInfo(userID : string) : void
+ doPost(HttpServletRequest,HttpServletResponse)
ItemView
- Button : Button
- TextField : TextField
- Frame : Frame
+ clickItem(itemtid) : void
WishlistController
+ doPost(HttpServletRequest,HttpServletResponse)
PaymentController
+ doPost(HttpServletRequest,HttpServletResponse)
+ acceptPayment(userid,itemid) : bool
Message
-messageid : string
-messagecontent : string
-sender : string
-receiver : string
-timestamp : date
+ getMessage(messageId : string) : string
+ getMessage(userid : string) : string
BidView
- button : Button
- textField : textField
- Frame : Frame
+ clickConfirmBid(userid, bidid, bidprice) : void
+ clickSellerInformation(userid) : void
+ clickBidInformation(bidid) : void
UserView
- button : Button
- textField : TextField
+ clickLogin(User) : void
+ clickRegister(User) : void
+ clickUpdateProfile(User) : void
+ clickDeleteUser(User) : void
+ clickUserInfo(User) : void
WishlistView
- button : Button
- textField : TextField
+ clickItemButton(itemd) : void
+ clickWishlistButton(userid) : void
PaymentView
cardinfo : TextField
barterindo : TextField
button : Button
+ clickPayByCard(cardinfo) : void
+ clickPayByBarter(barterinfo) : void
+ clickProceed(userid,itemid) : void
MessageController
+ doPost(HttpServletRequest,HttpServletResponse)
+ getMessageList (userId : string) : string
+ sendMessage(userId , messageId) : bool
+ notifyBuyer(userid,messageid) : void
MessageView
- button : Button
+ clickMessage(userId) : void
+ clickSelect(messageId) : void
+ clickNotify(buyerId): void

New Class Diagram:

https://github.com/avra0601/CSCI-5448-OOAD-project/blob/master/Class_diagram_part3.jpeg


Legend
Observer Design Pattern
Factory Method Design Pattern
New Classes Added
Address
LoginView
Category
Buyer
Seller
LoginController
Admin
UserFactory
RegistrationView
Bid
BidView
Item
ItemView
UserView
User
RegistrationController
BidController
Payment
Wishlist
WishlistView
ItemController
UserController
PaymentView
PaymentController
WishlistController
Message
MessageView
MessageController
<<interface>> ItemObserver
<<interface>> ItemSubject