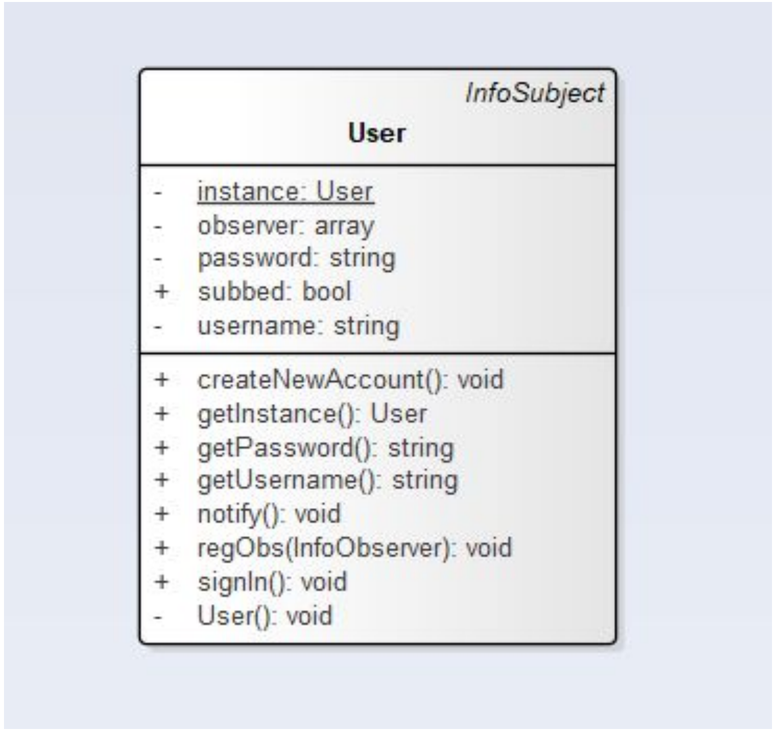
CIS 476 Term Project

Hayden Johnson
Matthew Haloostock

Singleton



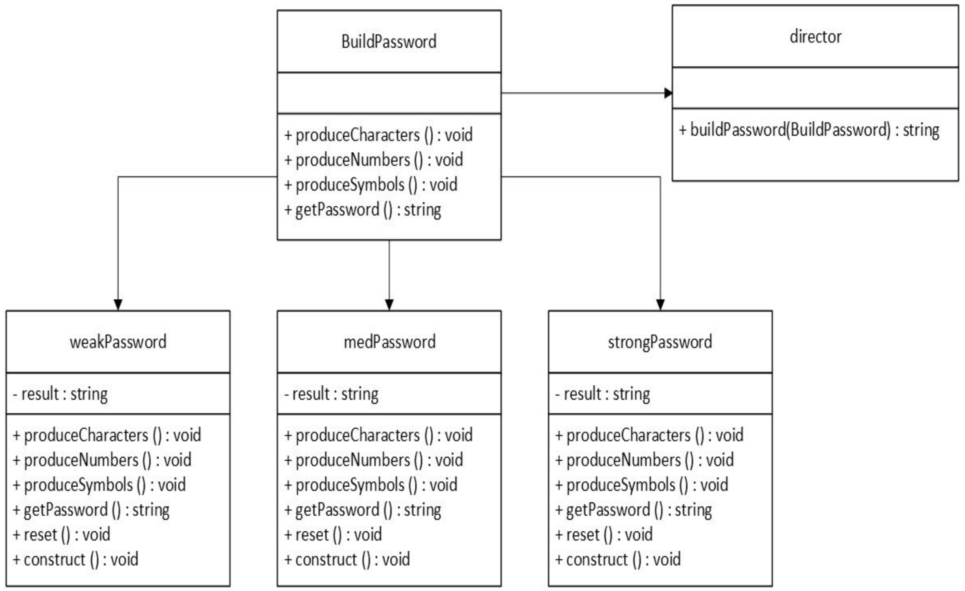
The singleton pattern is used for making sure only one instance of a variable exists at a time.

The User class is used for keeping track of the user across all pages.

The user can call User::getInstance() at any page and obtain the username and password of said user.

If one doesn't exist, (such as when first logging in) it will create one.

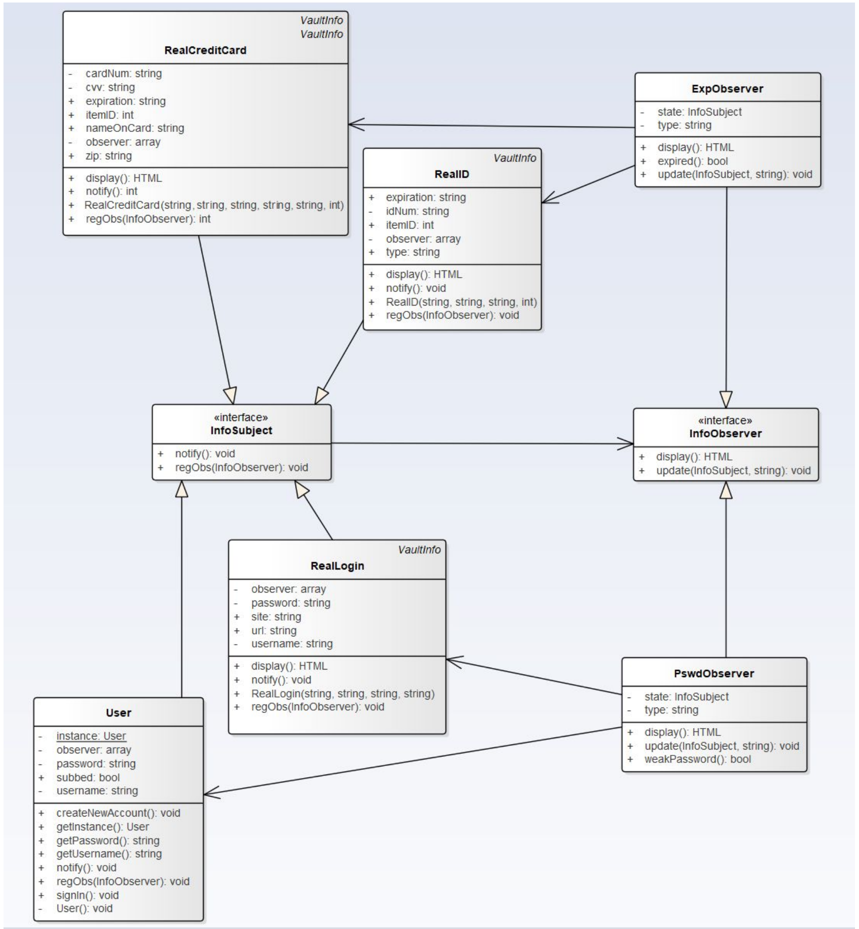
Builder


BuildPassword
+ produceCharacters () : void
+ produceNumbers () : void
+ produceSymbols () : void
+ getPassword () : string
director
+ buildPassword(BuildPassword) : string
weakPassword
- result : string
+ produceCharacters () : void
+ produceNumbers () : void
+ produceSymbols () : void
+ getPassword () : string
+ reset () : void
+ construct () : void
medPassword
- result : string
+ produceCharacters () : void
+ produceNumbers () : void
+ produceSymbols () : void
+ getPassword () : string
+ reset () : void
+ construct () : void
strongPassword
- result : string
+ produceCharacters () : void
+ produceNumbers () : void
+ produceSymbols () : void
+ getPassword () : string
+ reset () : void
+ construct () : void

The builder pattern is used for building objects when those objects have specific requirements depending on the context.

In our program, the context is for when the user asks for an auto generated password or varying degrees of security.

Observer



The Observer pattern is used when you need an object to keep track of another object’s state and handle its data accordingly.

We have an interface for both subject classes and observer classes.

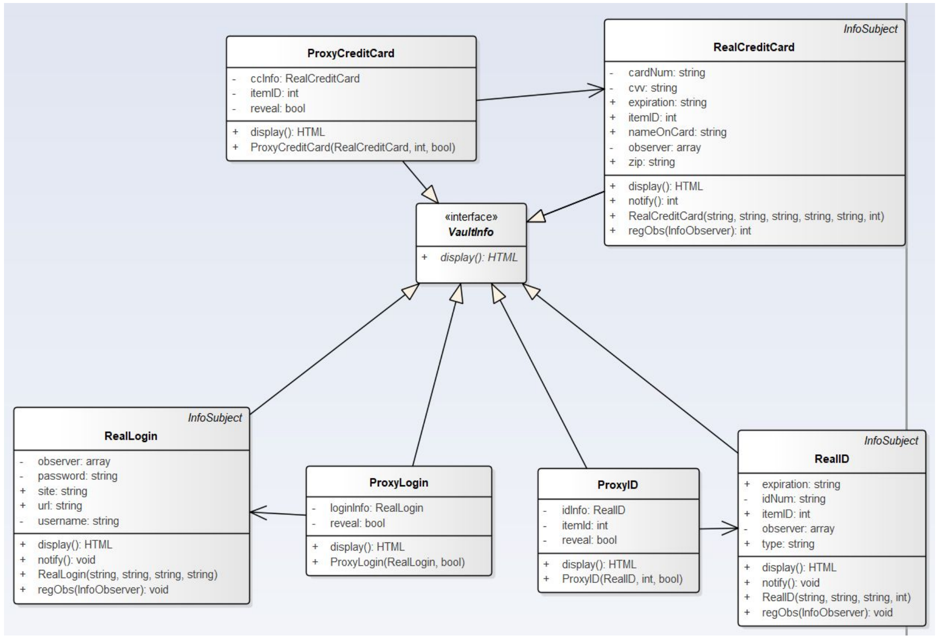
Our concrete subjects are User, RealLogin, RealCreditCard and RealID while the observers are PswdObserver and ExpObserver

PswdObserver notifies the user when there is a weak password, while ExpObserver notifies the user whenever there is an expired credit card or ID item.

Notifications are called when a concrete subject is instantiated since we are using php. Everytime an update happens, the classes are re-instantiated.

(Note: UML updated error from presentation for concrete observer attributes - state: InfoObject added to both)

Proxy



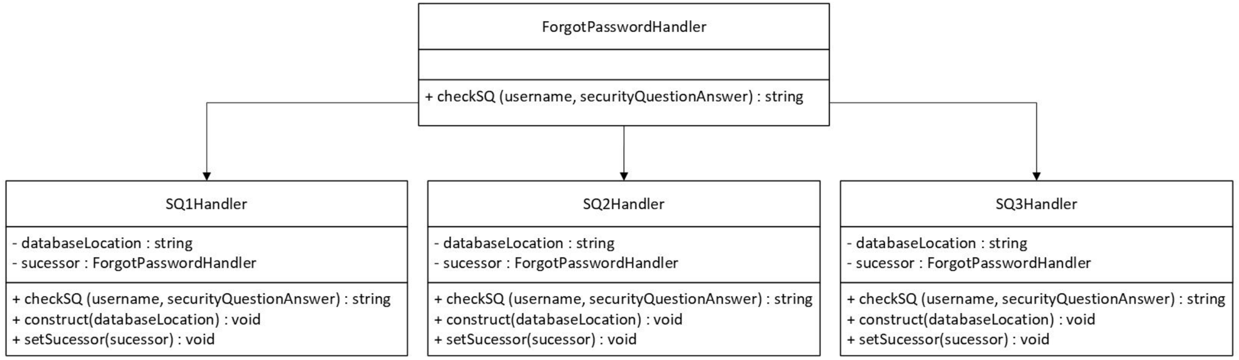
The Proxy pattern is used to hide real objects until they are needed.

In this project we used an interface to make sure that the proxy and real objects all have a display function.

The proxy classes store a real object so that access to the real object is controlled through the proxy (ProxyLogin stores RealLogin, ProxyCreditCard stores RealCreditCard and ProxyID Stores RealID.)

The proxy classes will display or hide sensitive information at the user's request.

Chain of Responsibility


ForgotPasswordHandler
+ checkSQ (username, securityQuestionAnswer) : string
SQ1Handler
- databaseLocation : string
- sucessor : ForgotPasswordHandler
+ checkSQ (username, securityQuestionAnswer) : string
+ construct(databaseLocation) : void
+ setSucessor(sucessor) : void
SQ2Handler
- databaseLocation : string
- sucessor : ForgotPasswordHandler
+ checkSQ (username, securityQuestionAnswer) : string
+ construct(databaseLocation) : void
+ setSucessor(sucessor) : void
SQ3Handler
- databaseLocation : string
- sucessor : ForgotPasswordHandler
+ checkSQ (username, securityQuestionAnswer) : string
+ construct(databaseLocation) : void
+ setSucessor(sucessor) : void

Chain of Responsibility patterns are when there are multiple ways to handle a problem. Ideally, if the invoked process cannot handle the problem, it will invoke a successor. If the successor cannot handle the problem, it will invokes its successor, and so on.

In our program, when a password is forgotten and the user submits their master password recovery questions, the array containing the answers will make its way down a chain of security question handlers before returning the result.

Contributions

Hayden - Builder, Chain of Responsibility, web application design, UML class diagrams, report

Matthew - Observer, Proxy, Singleton, UML class diagrams, report