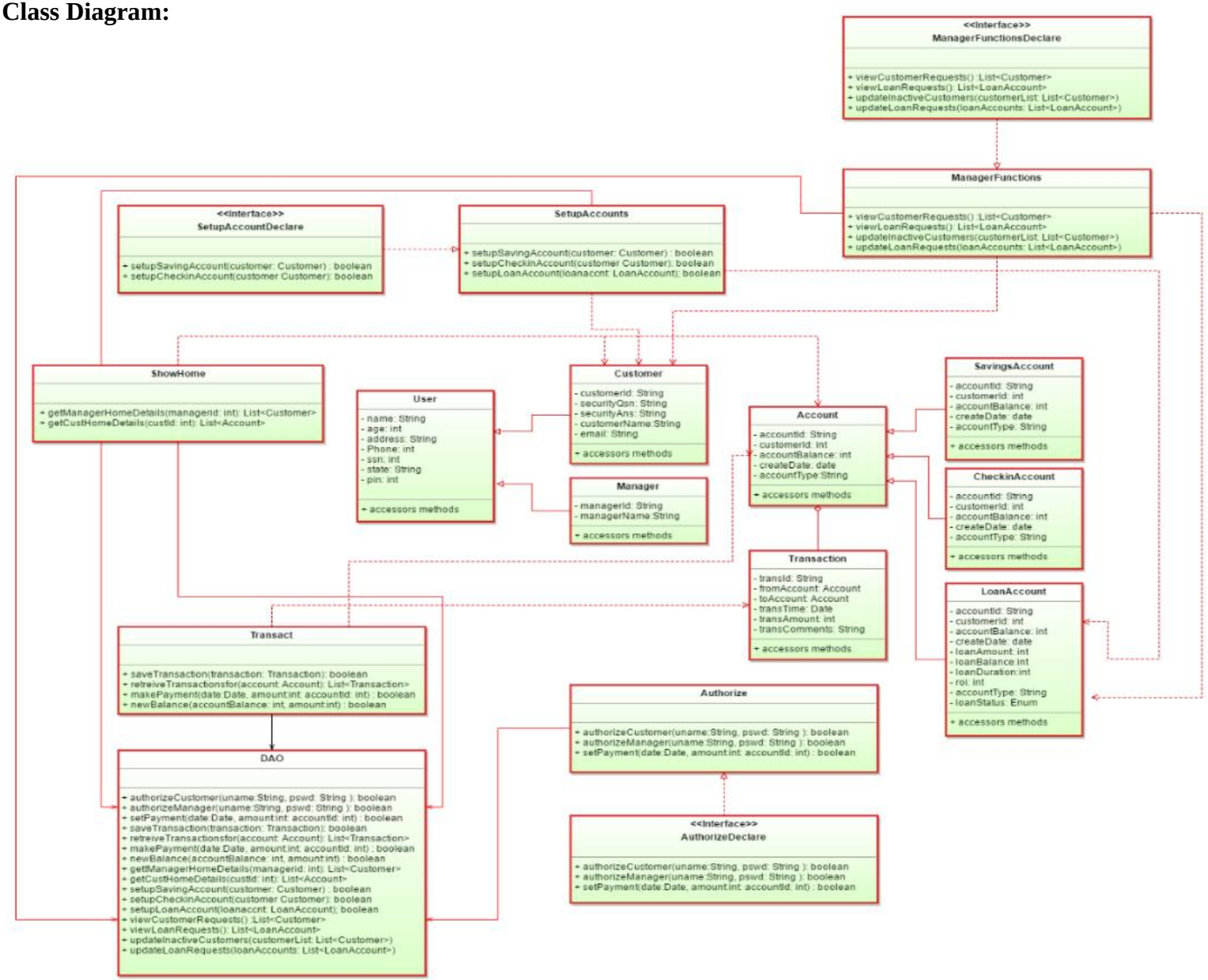
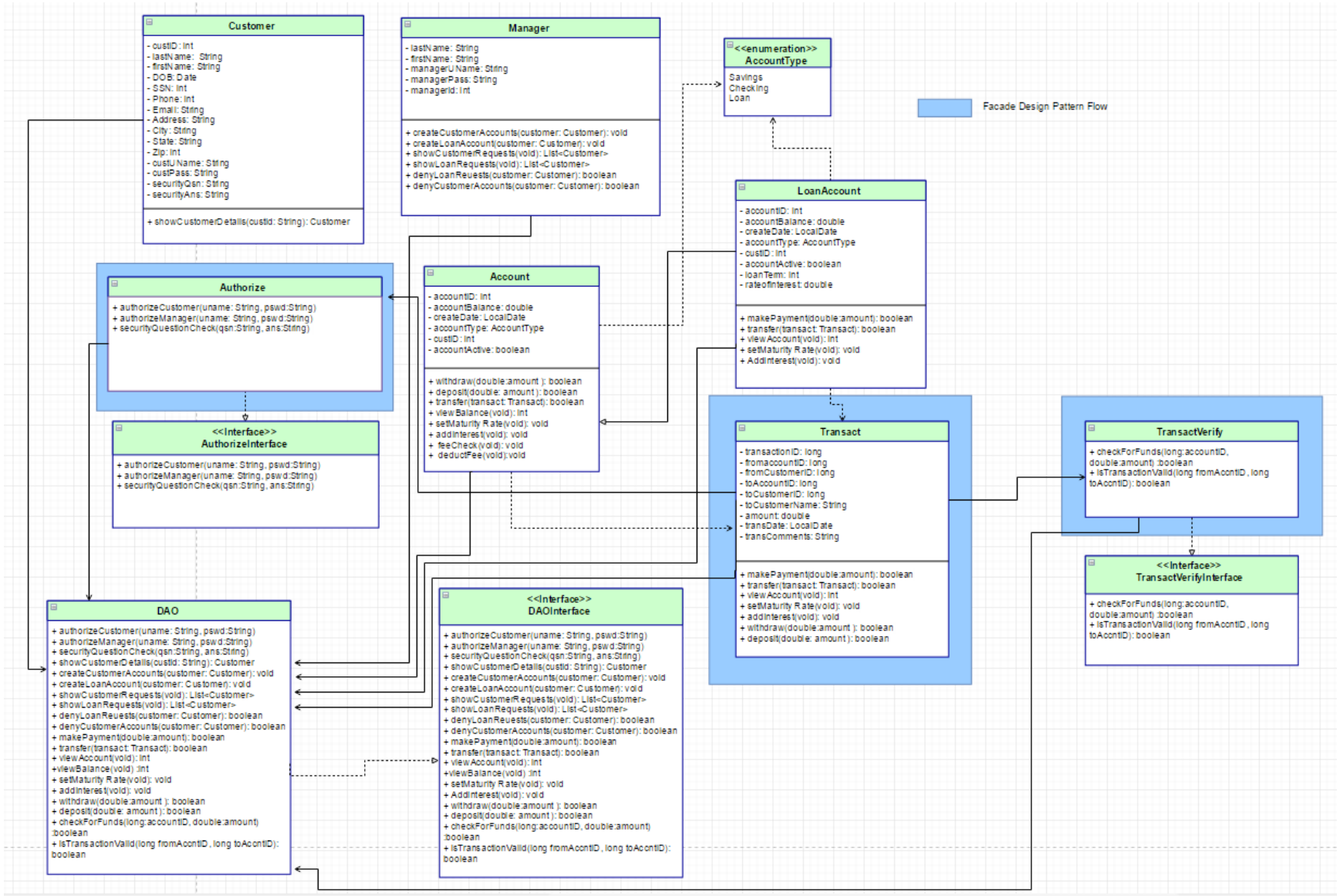
Part 2 Class Diagram:



Refactored Class Diagram Implementing the Facade Design Pattern:



Design Pattern Applied:

We applied the facade design pattern to our project. Users encounter the facade while performing a transaction (making a payment, transferring, withdrawing, or depositing funds). The Transact class acts as a facade to authorize a transaction based on a user's security question answer (Authorize class) or confirm the transaction (TransactVerify class) for various transaction types.

Refactoring Applied:

After much thought, we applied many changes to our original class diagram.

- Our SavingsAccount and CheckingAccount classes inherited their data types and methods from their parent class, Account. However, they only contained accessor methods inside of their classes. Noticing this bad practice, we were able to remove the SavingsAccount and CheckingAccount classes and only use the Account class by creating and <<enumeration>> named AccountType.
- The Account class now contains the methods withdraw, deposit, transfer, viewBalance, setMaturity, addInterest, feeCheck, and deductFee.
- The loanAccount class still inherits from the Account class. However, it no longer contains only accessor methods. It now contains the methods makePayment, transfer, viewAccount, setMaturity, and addInterest.
- We were able to remove the parent class User because the Manager child class didn't need to inherit its variables and methods. Instead we kept the two classes, Customer and Manager, and removed their parent, User.
- The Manager class now contains the methods: createCustomerAccounts, createLoanAccount, showCustomerRequests, showLoanRequests, denyLoanRequests, and denyCustomerAccounts.
- The Customer class now contains the methods showCustomerDetails.
- We created a new, needed class TransactVerify with the methods checkForFunds and isTransactionValid and the TransactionVerifyInterface <<Interface>>.
- We implemented the Transact class and Authorize class with the class TransactVerify for the facade design pattern.
- We changed the methods that the Transact class uses.
- We removed setPayment method from Authorize class and added the securityQuestionCheck method.
- We kept our AuthorizeInterface <<Interface>> .
- We kept our data access object or DAO class and the DAOInterface <<Interface>> to provide specific data operations without exposing the details of the database.

By doing this refactoring we were able to remove unnecessary classes, interfaces and many methods.

Classes Removed: ShowHome, User, SavingsAccount, CheckingAccount, ManagerFunctions, SetupAccounts, Transaction

Interfaces Removed: SetupAccountDeclare and ManagerFunctionsDeclare