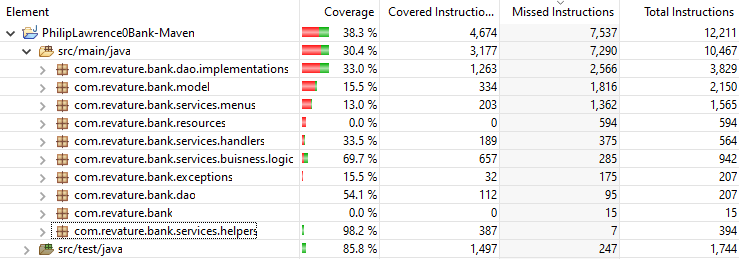
Project zero

## JUnit Testing

### Overall

Covered 4,674 Instructions of 12,211 (38.3%)

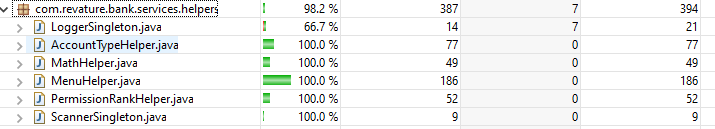


### Services

#### Critical

Methods where errors will cause security vulnerabilities or cascading null reference pointers throughout the code.

Covered 387 of 394 (98.2%)

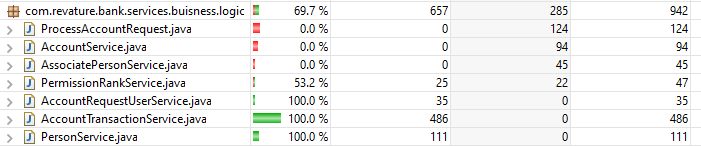


The missing commands of the LoggerSingleton come from the Access Denied log added the final day of the sprint. As this is mission critical (an error here would leave us blind to an employee trying to force their way past our security), it needs to be tested in the next sprint, even though it is very simple and durable.

#### Important

Methods that perform important business logic and need to be properly deployed as failure here can allow invalid data to be entered into the database.

Covered 657 of 942 instructions (69.7%)

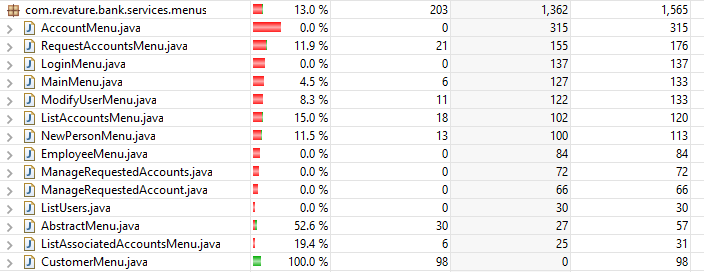


These functions are important and need full testing before deployment. The one hiccup during presentation would have been detected had ProcessAccountRequest been fully tested, however extreme time constraints led this function to being written functional first (with a small error) instead of TDD.

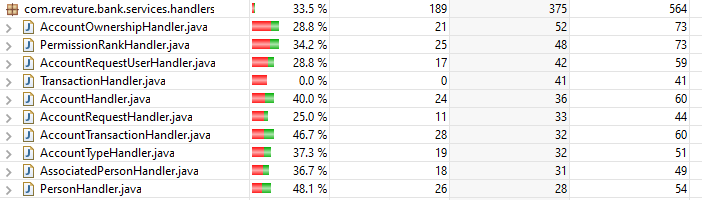
#### Luxury

Methods that improve user experience. Mistakes here will make sections of the program unavailable, but will not compromise data integrity

392 of 2129 instructions (18.4%)



Menu Functions are AbstractMenus that contain factories returning new menus used for navigation within the program.



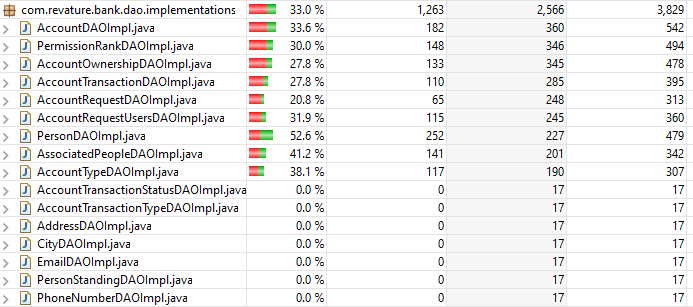
Handler functions exclusively forward data back and forth to the DAO allowing a place where the code can choose to implement a handler with a different DAO if some data needs to be persisted in a separate database.

### DAO

1375 of 4036 Instructions (34.1%)



DAO utilities hold the connection method as well as lazy singleton getters for each DAOImpl to provide faster implementation.



The DAOImpl at the bottom with 0% coverage are for stretch goals that will add additional functionality that goes beyond the scope of this sprint.

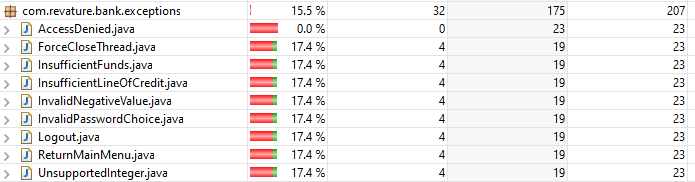
### Models

334 of 2150 Implementations (15.5%)



### Exceptions

32 of 207 Instructions (15.5%)



### Unimplemented Stretch Goals

0 of 594 Instructions (0%)



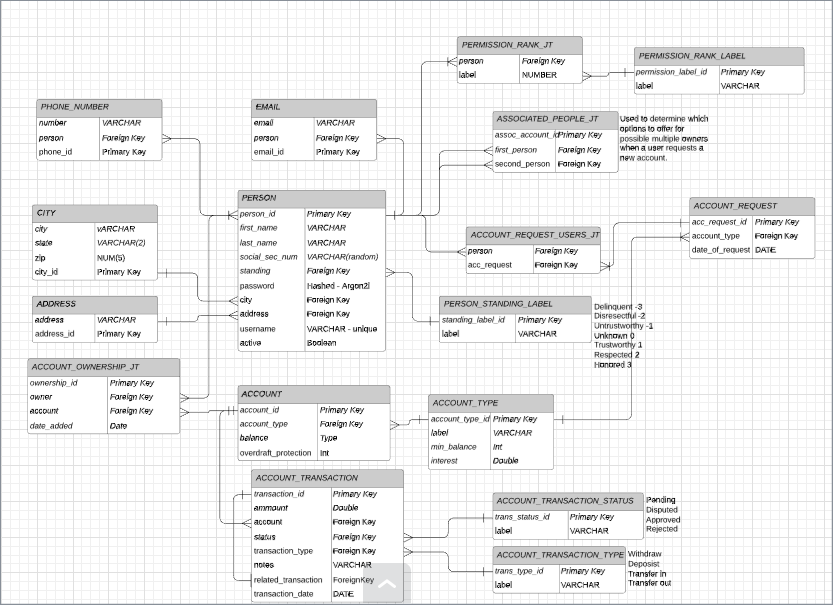
### Incidental Coverage of Test Classes

1497 of 1744 Instructions (85.8%)



## SQL

### ERD



Full Version: <https://www.lucidchart.com/invitations/accept/dbc26236-0c8b-4574-b6a3-bdf2dd0e3b96>

### Tables

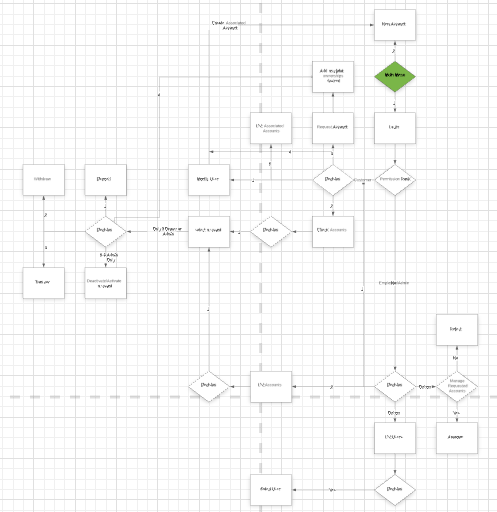
1. ACCOUNT
2. ACCOUNT\_OWNERSHIP\_JT
3. ACCOUNT\_REQUEST
4. ACCOUNT\_REQUEST\_USERS\_JT
5. ACCOUNT\_TRANSACTION
6. ACCOUNT\_TRANSACTION\_STATUS
7. ACCOUNT\_TRANSACTION\_TYPE
8. ACCOUNT\_TYPE
9. ADDRESS – only in SQL, not implemented in Application
10. ASSOCIATED\_PEOPLE\_JT
11. CITY – only in SQL, not implemented in Application
12. EMAIL – only in SQL, not implemented in Application
13. PERMISSION\_RANK\_JT
14. PERMISSION\_RANK\_LABEL
15. PERSON
16. PERSON\_STANDING – only in SQL, not implemented in Application
17. PHONE\_NUMBER – only in SQL, not implemented in Application

### Procedures

1. create\_account
2. create\_account\_ownership\_jt
3. create\_account\_request
4. create\_acc\_tran
5. create\_acc\_tran\_status – only in SQL, not called in Application
6. create\_acc\_tran\_type – only in SQL, not called in Application
7. create\_account\_type
8. create\_account\_req\_users\_jt
9. create\_address – only in SQL, not called in Application
10. create\_associated\_people\_jt
11. create\_city – only in SQL, not called in Application
12. create\_deposit
13. create\_withdraw
14. create\_transfer
15. create\_email – only in SQL, not called in Application
16. create\_person
17. create\_standing – only in SQL, not called in Application
18. create\_rank – only in SQL, not called in Application
19. create\_phone – only in SQL, not called in Application

## Functionality

### Map



Full Version: <https://www.lucidchart.com/invitations/accept/32d06bc5-cda0-4b37-9790-a18d4ab0fccc>

### Customers

* Create login user
* Set username (on creation)
* Set password (on creation)
* Set first name
* Set last name
* Set random SSN
* Request new account
* Deposit into owned account
* Withdraw from owned account
* Transfer from owned account
* Associate login user with another login user (to allow requesting joint account)
* Request joint account
* Add user to existing owned account

### Employees

* View user information
* View account information
* Approve pending accounts

### Admins

* View user information
* View account information
* Withdraw from any account
* Deposit into any account
* Transfer from any account
* Add any user to any account
* Deactivate any account
* Activate any account

### Validation

* Full sanitization of user input to either
  + Integer
  + Positive Integer
  + Token (one word alphanumeric)
  + Positive Double (two-digit decimal) – used for user views and transactions
  + Positive Double (five-digit decimal) – used for internal tracking of fractional cents for interest
* Withdraw and Transfer limitation
  + Limited by account\_type’s minimum balance
  + Limitation expanded by account’s overdraft protection