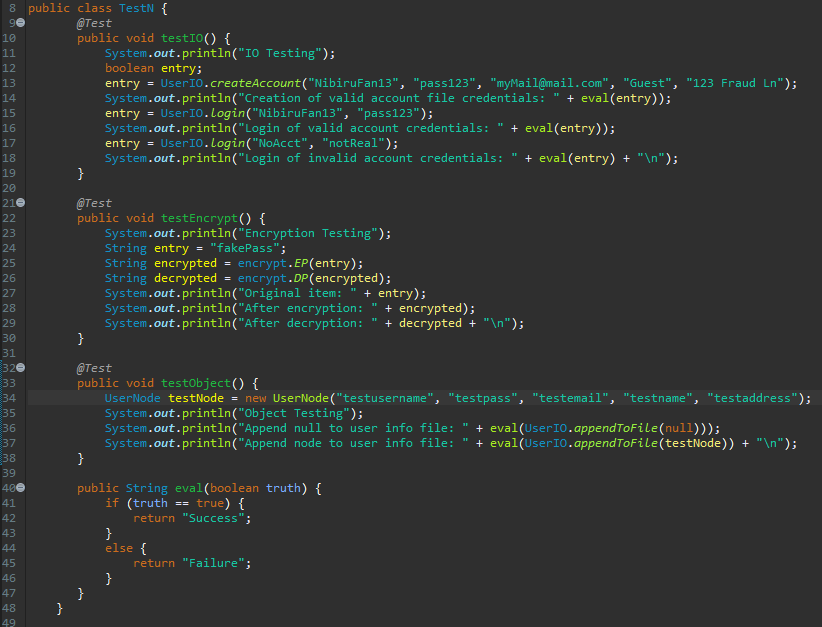
**Software Testing Plan for Nibiru**

**Unit Testing**

The project will utilize Top-down integration testing. The logic behind this decision is that the main control unit of the site can be used as a platform for the other components that will also need to be tested and evaluated. Testing will start with the subordinate “stubs” being in place of the other modules below the control. Over time we will replace these with the actual components of the software. For our control, we will be using the Account class as out test driver. The User, Message, Item, and Transaction modules will be taking the place of the stubs under the control. System will also be tested as a utility component. Using this unit testing strategy is ideal because as new features and modules are added to the software, more tests can be conducted as the project timespan continues. As each set of components completes their testing, other modules will take their place.

**Validation Testing**

Being that the project is web based, validation testing will be focused primarily on beta testing. A test release will be made available after internal testing at the developer site. This release will only include the main features of the main site and all of its utilities. As features are added and changes are made, newer releases of the beta build will be distributed after waves of unit testing until the consumer ready release.

**JUnit Test Cases**

**Testing Input / Output**

Testing method testIO() will be used to determine if the user can make and account given all the credentials necessary to build the account. This method will also test if the user can login with both valid and invalid credentials. The outputs will display if the operations were successful. It is ideal to have the account creation be successful when there is no existing file and to be a failure when the file already exists. Valid credential login should always be successful and invalid login should always fail.

**Testing Encryption/Decryption**

Method testEncrypt() will isolate the encryption and decryption utilities to test if a given string will encrypt after calling the encrypt function. Next, the return of the encrypt will be used as a parameter in the decrypt function. Both the returns of the two methods will be printed as well as the original string. The decrypt should match the original while the encrypt should be unique.

**Testing Object (UserNode)**

Method testObject() will determine if the creation and appending of a UserNode is successful. The append method will be tested will null and a new instance of the node that is constructed with test values for the arguments. The output will show if the test fails or succeeds. Obviously, the null test case should always fail and the valid node should succeed.

**Results and Analysis**

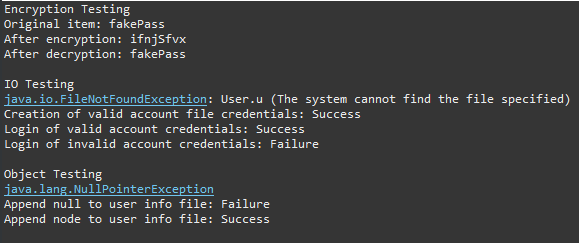
The results of the test cases show that the methods are rejecting the improper parameters and accepting the proper ones. In figure 1, we see that null nodes cannot be created or appended to the file and that valid accounts can be made and logged into. Encryption and decryption are working properly given valid input as well.

Figure : First run of tests (creates new user file)

After the first run of these cases you will notice that the user file has now been created. Running the test again shows that the creation of a valid account is now a failure. This is because the sample is not designed to have more than one account file. The method is meant to return false if this file is already in existence so the failure of this case is favorable.

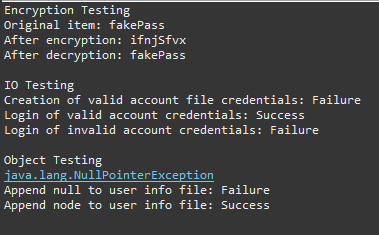
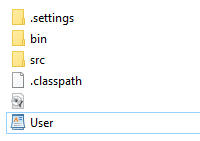
This evaluation the methods and their parameters will be the basis for testing the components of existing modules and those to come later. Using this strategy in the top-down integration method, we can test the components in cycles as they are added to the site. As this is representing a website that will be adding features throughout its lifespan, this will be ideal for growth and evolution.

Figure : Run of test cases with user file already created

Figure : User file is now created

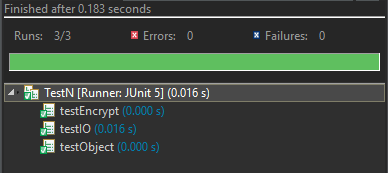


Figure : Test case success status